ANALYZING CROSS-LAYER INTERACTION IN OVERLAY NETWORKS

A Thesis
Presented to
The Academic Faculty

by

Srinivasan Seetharaman

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
College of Computing

Georgia Institute of Technology
December 2007

ANALYZING CROSS-LAYER INTERACTION IN OVERLAY NETWORKS

Approved by:

Dr. Mostafa Ammar, Advisor
College of Computing
Georgia Institute of Technology

Dr. Constantine Dovrolis
College of Computing
Georgia Institute of Technology

Dr. Nick Feamster
College of Computing
Georgia Institute of Technology

Dr. Markus Hofmann
Director, Multimedia Networking
Bell Labs, Alcatel-Lucent

Dr. Ellen Zegura
College of Computing
Georgia Institute of Technology

Date Approved: 06 September 2007

To my family, for their support and encouragement.

ACKNOWLEDGEMENTS

This thesis would not have been possible without the support and encouragement of people too numerous to enumerate, but I will try to acknowledge their efforts anyway.

First and foremost, I thank my thesis advisor, Mostafa Ammar, without whose patient guidance and limitless support, I would not have completed this journey. I can only wish I turn out to be half as good an advisor as he is. I would also like to thank Constantine Dovrolis, Nick Feamster and Ellen Zegura for their valuable comments and insights, beginning with my first NTG seminar presentation. I thank Jim Xu for his boundless humor and constant counseling. I am thankful to my external committee member, Markus Hofmann, for his valuable guidance and for giving me the opportunity to spend a very enjoyable summer at Bell Labs. Thanks also to Volker Hilt of Bell Labs for being a great mentor, collaborator and friend.

This thesis research was made a lot easier by the facilities at the College of Computing and, of course, by my lab mates. Without their constant support, my life at Georgia Tech would have been a lot less pleasant. I thank them all. Special thanks goes to Mukarram for all the interesting discussions and the critical reviews; Anirudh for being an excellent collaborator and for providing a "Trance" work environment; Amogh and Ruomei for being ideal medium for bouncing ideas; Murtaza for his company in the lab at vague unearthly hours.

The last five years have been very rich and enjoyable owing to the presence of two special people - Avanti and Badri. Their constant moral support, monetary loans, ridiculing, cooking, conversations, wit, humor, magnanimity and general presence have kept me going. They mean more to me than words can tell. Then there's Sridhar, who I never really saw as my lab-mate, but as a mentor both academically and personally. His innumerable counseling sessions helped shape my career. I also plagiarized much of this acknowledgment from him. I can almost hear him laughing and calling me "Muttaal".

I acknowledge the support of my friends Anu, Dounia, Farzan, Millo, Nisarga, Nupur, Pradnya and Sumegha for making life easier and fun in Atlanta. They are my family here and have been there through all my ups and downs. They have been gracious enough to forgive me for avoiding them during deadlines. I thank Amisha and Chirag for their short, yet substantial presence in my life. It has been fun spending time with them. Thanks are also due to my friends Manoj, Nigamanth and Vinod for making me feel at home in Ohio and being the good friends they are. My Atlanta friends Aru, Deepak, Gayathri, Girija, Karthik, Rajesh and Shilpa have also been there as much as possible, and ensured that I never become too much of a lab-rat. I am deeply indebted to them for their support. I would also like to acknowledge all that I have got from my association with the Association for India's Development (AID). It has exposed me to the real world and inculcated qualities that can never be taught.

None of these experiences would have occurred if not for the perpetual support of my parents, brother Mahadevan and sister-in-law Archana. During my 7 years of graduate study, never once did they ask "When are you graduating?". Their unconditional belief in my abilities still amazes me. I dedicate this thesis to them.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

Overlay networks have recently gained popularity as a viable alternative to overcome functionality limitations of the Internet (e.g., lack of QoS, multicast routing). They offer enhanced functionality to end-users by forming an independent and customizable virtual network over the native network. Furthermore, they are being widely promoted as a potential architecture of the future Internet in the form of *network virtualization*, where multiple heterogeneous virtual networks may co-exist on top of a shared native network. The prominent characteristic in either context is that routing at the overlay layer operates independent of that at the underlying native layer.

There are several potential problems with this approach because overlay networks are selfish entities that are chiefly concerned with achieving the routing objective of their own users. This leads to complex cross-layer interactions between the native and overlay layers, and often tends to degrade the achieved performance for both layers. As overlay applications proliferate and the amount of selfish overlay traffic surges, there is a clear need for understanding the complex interactions and for strategies to manage them appropriately. Our work addresses these issues in the context of *service overlay networks*, which represent virtual networks formed of persistent nodes that collaborate to offer improved services to actual end-systems. Typically, service overlays alter the route between the overlay nodes in a dynamic manner in order to satisfy a selfish objective. The objective of this thesis is to improve the stability and performance of overlay routing in this multi-layer environment.

We investigate the common problems of functionality overlap, lack of cross-layer awareness, mismatch or misalignment in routing objectives and the contention for native resources between the two layers. These problems often lead to deterioration in performance for the end-users. This thesis presents an analysis of the cross-layer interaction during fault recovery, inter-domain policy enforcement and traffic engineering in the multi-layer context. Based on our characterization of the interaction, we propose effective strategies that improve

overall routing performance, with minimal side-effects on other traffic. These strategies typically 1) increase the layer-awareness (awareness of information about the other layer) at each layer, 2) introduce better control over routing dynamics and 3) offer improved overlay node placement options. Our results demonstrate how applying these strategies lead to better management of the cross-layer interaction, which in turn leads to improved routing performance for end-users.

CHAPTER I

INTRODUCTION

Overlay networks have recently gained attention as a viable alternative to overcome functionality limitations (e.g., lack of QoS, difficulty in geo-positioning, multicast support) of the Internet. The basic idea of overlay networks is to form a virtual network on top of the native network so that a few specialized overlay nodes can be customized to incorporate complex functionality without modifying the underlying native routers. Typically, these overlays route packets over paths made up of one or more overlay links to achieve a specific end-to-end objective. Each intermediate overlay node is referred to as a *relay* and the forwarding operation at each hop is referred to as *relaying*.

These overlay networks can be of different classes based on their specific purpose. Some common forms are peer-to-peer (P2P) overlays, content delivery networks (CDN), and service (infrastructure) overlays[29]. This thesis is particularly focused on the performance and stability observed in service overlays[39], which are overlays formed to offer a value-added network service to actual end-systems. Service overlays are of particular interest to us because they act like *networks* and in many respects try to mimic the functionality designed within the native network. The member nodes of this overlay are persistent and route traffic between one another.

Third-party service providers can use these overlays to offer services, currently unavailable in the native network, to their customers. Examples of such services includes multicast (e.g., Narada[28], Overcast[78]), optimized paths (e.g., RON[6], Detour[51], X-Bone[155], Brocade[175]), customized forwarding (e.g., I3[147], Scattercast[25]), quality of service (e.g., OverQoS[149], SON[39]) and security (e.g., SOS[85], DynaBone[156]). Furthermore, they display great promise as the central architecture for the future Internet[141], thereby elevating their importance.

Typically, service overlay networks are deployed by means of IP-tunneling and they

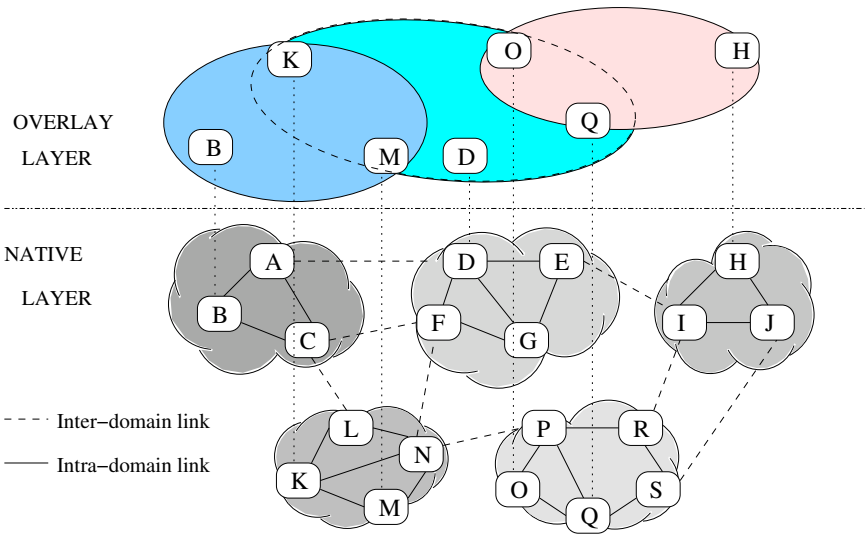


Figure 1: Illustration of the multi-layer architecture in service overlays.

maintain an overlay routing table that is independent of underlying native network[6, 51, 28, 155]. Figure 1 illustrates the structure of a typical service overlay. Each *overlay link* represents the direct native route between two overlay nodes, which in turn comprises one or more native links, and each *overlay path* is made up of one or more overlay (virtual) links. This overlay path represents the end-to-end route taken by the overlay application traffic. Based on the specific routing objective, the native and overlay layers pick the exact sequence of element (nodes and links) the traffic must traverse in that particular layer. The traffic between nodes K and Q at the overlay layer is determined by routing at that layer, with the path traversed by each overlay link determined by native IP routing protocols. A second dimension of complexity is brought into picture when multiple overlay networks are overlaid on the same substrate network, with varying levels of overlap.

In this chapter, we present an overview of issues involved in the operation of such a service overlay network and enumerate specific problems that we investigate in this thesis. Further, we present a formal organization of the overall thesis.

1.1 Cross-Layer Interaction

Service overlays have the capability to sense changes in the underlying native network and dynamically adapt their routing tables to changing network conditions such as node/link

failure, congestion, and increasing demands. This is a selfish approach aiming to offer enhanced routing performance to the overlay network's traffic. Furthermore, the native and overlay layers operate independent of each other in order to achieve a particular objective. For these reasons, the cross-layer interaction between the overlay network and the underlying native network is interesting in its own right.

We are specifically interested in 1) how the routing protocols interact across the multiple network layers? 2) how visible is the operation and parameters of one routing layer to the other? 3) how best to interface between the two routing layers? and 4) how the objectives of the routing protocols at each layer align or interfere with each other? The general theme of our work is to investigate the potential impairments caused when the two layers operate independent of each other, and to develop ways for improving route optimality and network stability.

We evaluate the functioning of overlay networks in several scenarios to obtain insights into the cross-layer interaction. First, we analyze the cross-layer interaction during recovery of link failures that originate at the native layer. Second, we characterize the violation of inter-domain policies specified at the native layer and the deterioration in user experience when the native layer enforces these policies through filtering. Third, we investigate the interaction between overlay routing and traffic engineering in a resource-constrained native network, each entity aiming to achieve a different local objective. Fourth, we investigate the impact of BitTorrent file-sharing on the inter-domain traffic variation. In all four scenarios, we see a common characteristic that the native and overlay layer neither cooperate nor exchange information between each other. This can be modeled as a two-player non-cooperative repeated (recurring) game without a stable operating point. This deteriorates the performance achieved by each layer. Lastly, we investigate ways in which the native layer can cooperate with overlay services, without affecting the non-overlay traffic.

The main insight of this thesis is that the deterioration in the system performance is caused by the following three forces:

- **Functionality Overlap:** In a multi-layer routing scenario, each layer often conducts most of the standard routing functions – like link verification, path computation,

policy enforcement, route optimization, topology design, load balancing – causing a functionality overlap. Replicating the core functions in multiple layers can lead to route flaps and suboptimal routes (because each layer tends to undo the operations of the other layer), wasteful efforts (caused by an unawareness of the exact layer offering the best service for a particular scenario) and undesired overhead[138].

- **Coexistence and Resource Contention:** In the multi-layer environment, overlays compete for native network resources with other overlays[83] (at the same or different level) and with native traffic[82]. While sharing of network resources among native flows has received significant attention in previous research, this multiple-layer environment is considerably more complex. This is especially true because of the fact that overlay layers are generally performance-aware and may deploy routing protocols that attempt to improve performance as network conditions change.

- **Mismatching Objective:** A conflict in objective between the individual routing layers and a misalignment of objective (because of varying parameters and environment) can exacerbate the problems of network instability. We show that the selfish behavior of overlay routing has significant conflict with the traffic engineering efforts of the ISP[139] and with the enforcement of inter-domain policies[137].

In the current Internet, we see a constant power struggle between the native layer, which is optimized for a broader range of users, and the overlay layer, which is solely interested in improving its own routes. As overlay applications proliferate, it is highly likely that the amount of selfish overlay traffic will experience significant growth in the future. In such scenarios, there is a crucial need for means to mitigate cross-layer conflict. The objective of this thesis is thus to *analyze and mitigate this cross-layer conflict*. We addresses this need using a combination of *layer-awareness* (awareness of information about the other layer's existence) and a strategic redesign of the overlay topology. The general idea of the solutions is to help retain the routing advantage offered by the service overlay networks, with minimal side-effects on the other traffic (both from legacy and other overlay applications). The solutions we propose are fundamentally simple and effective. Further, we present a

framework for cross-layer cooperation in an effort to achieve the best possible performance for overlay services by *modifying* the operation of the current native layer. This modification helps resolve the impasse in the progress of the current Internet[141].

Although much of the impairments of cross-layer interaction are seen in the case of persistent (infrastructure) overlays, we expect similar dynamics in the case of other *performance-aware* overlays, like Bittorrent, that adapt their functioning and routes based on measurements or estimates of the state of the underlying native network. This is a selfish approach that can have similar impact on the operation of the underlying native network (like conflict with traffic). In this thesis, we investigate the impact of BitTorrent file-sharing on the native layer traffic management operations.

1.2 Summary of Research Objectives

We investigate the following five problems in this thesis, in order to obtain insights into the cross-layer interaction in service overlay networks:

Analyzing Interaction between Fault Recovery in the Native and Overlay Layers: We investigate scenarios where a dynamic routing protocol is employed both at the overlay layer and the underlying native layer, to adapt their respective routing tables to changing network conditions. In particular, we analyze the interaction between these two routing layers during the rerouting around failed links. We first study a *Dual Rerouting* scenario in which the two routing layers run completely independent of each other. We obtain an understanding of the effect of the various settings of routing protocol parameters on the packet loss, number of route flaps, and the optimality of the adopted overlay path. However, owing to the overlap of functionality between the two layers and the unawareness of the other layer's decisions, Dual rerouting tends to be sub-optimal in terms of the number of route flaps and the overlay path cost inflation. Based on that observation, we propose layer-aware schemes that helps us trade off longer path recovery times with improvements in route flapping and path cost inflation.

Analyzing Inter-Domain Policy Violations in Overlay Routes: We make the observation that routing in service overlay networks are quite capable of violating the commercial agreements between neighboring autonomous systems (AS) and the exit preference of each AS. We analyze a case study overlay network, constructed on top of Planetlab, to gain insights into the frequency and characteristics of the different inter-domain policy violations. Nearly 70% of the multihop overlay paths in our testbed violated the transit policies and nearly 87% of the paths violated the exit policies. This has serious implications (economic and load), thereby motivating the native ASes to detect and filter overlay traffic. We next investigate the impact of two types of overlay traffic filtering that aim to prevent these routing policy violations: *blind filtering* and *policy-aware filtering*. We show that such filtering can be detrimental to the performance of overlay routing. To attain a solution mutually agreeable to both the native and overlay layer, we propose a single *cost-sharing* framework that helps legitimize all native policy violations and allows the benefits obtained by the overlay to be directly related to costs incurred by the overlay service provider.

Analyzing Conflict between Overlay Routing and Traffic Engineering: Overlay routing is known to cause undesired instability in a network by operating in a selfish manner. The objectives of overlay routing, such as optimizing end-to-end latency, are often in conflict with the objectives of traffic engineering in the native layer, which is concerned about balancing load. We build on past research that has investigated the recurring non-cooperative interaction between overlay routing and traffic engineering, and develop strategies that improve the routing performance of a particular layer with incomplete information about the other layer. In our strategies, one layer acts as a *leader* that predicts the *follower*'s reaction and undertakes countermeasures to prevent future deterioration in performance; similar to the Stackelberg approach in game theory. For each layer, we propose two classes of preemptive strategies: *friendly* or *hostile*. We show that these strategies achieve near-optimal performance for the leader and increase the overall stability of the network.

Analyzing Cross-layer Interaction in BitTorrent Networks: The BitTorrent protocol, through its tit-for-tat based performance-awareness, behaves in a selfish manner and

causes each peer to frequently alter its routing decisions (peer and piece selection). This causes fluctuations in the load experienced by the underlying native network. We investigate the cross-layer interaction between BitTorrent file-sharing and inter-domain traffic management strategies. By using real BitTorrent traces and a comprehensive simulation with different network characteristics, we show that BitTorrent systems easily disrupt the load balance across inter-domain access links. Further, we show that traffic engineering is ineffective in managing the impact of BitTorrent file-sharing. We also show that locality-based traffic management and bandwidth throttling strategies work well in reducing the overall inter-AS traffic and the peak load on each inter-domain link, albeit at the expense of user experience or infrastructure costs, making them unattractive for deployment. To resolve this dilemma, we propose and analyze two BitTorrent strategies that are effective in reducing the impact of BitTorrent traffic and steering the system towards a mutually agreeable operating point.

Improving Cross-layer Cooperation: It is a widely accepted notion that the native layer of the current Internet has begun to stagnate in terms of the services offered. Consequently, overlay networks have gained attention as a viable alternative to overcome functionality limitations of the Internet. A common approach to overlay network design holds the native network inviolable, implying that the overlay has to take on all the tasks needed to provide the desired high-level services. This limits the performance achieved and can potentially overburden the overlay layer. To solve these problems, we envision that, as overlay applications proliferate, the native layer should gradually evolve to suit the overlay network requirements. We propose a framework for such an *overlay-friendly native network* (OFNN), which will cater to the overlay applications without compromising on the performance of the non-overlay applications. Further, we investigate the option of tuning the native layer parameters as a simple, yet feasible, OFNN approach.

1.3 Thesis Organization

Chapter 2 discusses existing work related to the topics in this thesis. The cross-layer interaction between dynamic routing protocols in the native and overlay layers is discussed

Table 1: Thesis Organization

	Attitude of native network		
	Oblivious	**Restrictive**	**Cooperative**
Intra-domain	Dynamic metric-based rerouting Chapter 3	Load-balancing, under finite resource limits Chapter 5	Overlay-friendly native network Chapter 7
Inter-domain		Policy constraints Chapter 4 Load-balancing Chapter 6	

in Chapter 3. Chapter 4 presents an analysis of the inter-domain policy violations caused by overlay routing and ways to mitigate them. Chapter 5 discusses strategies to resolve the cross-layer conflict and to insure improved performance for each layer. We present our analysis of the cross-layer interaction seen in the context of peer-to-peer networks in Chapter 6. Chapter 7 presents design guidelines for an overlay-friendly native network, which further improves the performance of service overlays. The contributions of this thesis and the recommend future work are summarized in Chapter 8.

Table 1 organizes the different pieces of work based on whether the work investigates cross-layer interaction in intra-domain or inter-domain context, and based on the level of interaction between the two layers. When the two layers operate independently and have minimal impact on the operations of each other, we term the interaction as *oblivious*. When the native layer reprimands certain decisions of the overlay layer, we term the interaction as *restrictive*. When the native layer offers network-level support to overlay services, we term the interaction as *cooperative*. This thesis analyses all these six combinations.

CHAPTER II

RELATED WORK

This chapter summarizes some of the work related to the cross-layer interaction issues in overlay networks. Specifically, there is past work in investigating the fault restoration in optical networks, in investigating the conflict between overlay routing and traffic engineering, in investigating the resource contention between multiple overlay networks, in investigating the cross-layer interaction in BitTorrent networks, and in investigating the overlay node placement alternatives. Any specific work that is not related to overlay networks, yet related to the specific native layer dynamics is discussed in the corresponding chapter.

2.1 Multi-layer Fault Recovery

The problem of rerouting around failures in a single layer is a well-studied problem in different types of networks[35, 109, 168]. Typical solutions suggest resource reservation for a backup path that will be activated in the event of a failure or reactive approaches that perform dynamic restoration. Each strategy varies in the success rate of recovery, bandwidth used and the recovery speed. There has been little work in the field of overlay network recovery. Backup path allocation schemes exploiting the high correlation of overlay links have been studied in [34]. A substantial work in the field of overlay layer survivability is the work on resilient overlay networks (RON)[6], which is an effort to make the overlay network robust in the face of native layer problems. RON uses application-specific probes to determine the best path for particular metrics and performs dynamic reconfiguration. All these strategies behave sub-optimally when there are two layers attempting to reroute around a failure. We investigate that issue in this thesis and prescribe ways to achieve a desirable tradeoff between recovery time and route stability.

2.1.1 Multi-layer Recovery in Optical Networks

Multi-layer fault recovery is a well-studied problem in optical networks[132, 58, 7, 49], where the IP layer is an overlay of the underlying WDM layer. In wavelength division multiplexing (WDM) networks, it has been pointed out that an IP-only recovery scheme can support more traffic load as the backup capacity is not wasted, and a WDM-only scheme has considerably shorter recovery time without guaranteeing maximum load support. Thus, it is non-trivial to determine the exact layer to use for recovery[58]. The protection across network layers (PANEL) project[7, 49] further investigated the interaction of the recovery mechanisms deployed in the different layers of the transport networks. Increased layer coordination was suggested in the form of regulating a hold-time[1] and recovery-ratio[2]. The PANEL project also showed that the lower layer can recover higher load at a faster pace, by recovering multiple higher layer connections simultaneously.

However, in the context of overlay networks where the two layers operate independent of each other, it is not feasible to deploy a coordinated recovery mechanism, making our problem all the more interesting. Furthermore, the cross-layer interaction in optical networks is unlike the interaction we anticipate in the multi-layered Internet owing to three reasons. Firstly, past research does not account for selfish independent operation of each layer. In the multi-layered architecture, each layer is driven by its perceived performance. Secondly, the network span of the overlay layer is limited and can vary dynamically based on user demands. Lastly, in the future Internet, there may be multiple virtual networks overlayed on the same substrate network, which is not true with past research.

2.2 Conflict between Overlay and Native Layers

Often overlay routing encounters a conflict in objective with other non-overlay (other overlay or legacy applications) routing efforts. This conflict is exacerbated in the presence of resource limitations at the native (substrate) layer. [100, 98] investigate the conflict in objective between routing protocols at the overlay layer and traffic engineering (TE) deployed

[1] Time allotted to lower layer to complete recovery.
[2] Amount to be recovered in each layer.

at the native layer. Their general conclusion is that the interaction causes sustained route oscillations and sub-optimal performance for both layers. By means of simulation, which uses both synthetic topologies and real tier-1 ISP topologies as input, [100] shows that the interaction between selfish routing in overlay networks and TE are unlike traditional equilibria analyses[88, 130]. [98] models the native link latency as a function of the load experienced by the link and derives a precise definition of the overlay and native routing objective. Using this model, they show that the two objective are in constant conflict and that there exists a Nash equilibrium point. This thesis builds on previous work to analyze exact scenarios which causes this conflict and presents ways to mitigate the resulting instability.

[80] investigate the cross-layer interaction in the context of P2P networks. However, they restrict their analysis to the impact on ISP cost. Further, they propose locality-aware solutions that optimize the cost incurred by an ISP when their subscribers download content from different peers. We address the issues of load and policy both in the context of service overlay networks and P2P networks.

To resolve the aforementioned conflict, the work in [173] uses a Stackelberg approach to pick appropriate overlay routes. However, they assume complete information of the follower and a M/M/1 cost function for the two layers. Our preemptive strategies do not make these assumptions, and considers conflicting objectives between the two layers, without requiring complete information about the other layer. Furthermore, they use a gradient projection search to obtain an approximate solution, which is locally optimal and closer to the initial point, in one iteration. In contrast, the strategies we propose arrive directly at the optimal choice within a few rounds, and is unrestricted by the original solution space.

2.3 Conflict between Coexisting Overlay Networks

There exists another dimension of interaction that impacts the overlay routing performance. It has been shown that overlays competing for native network resources with other overlays increase the network instability[83, 84]. As the number of coexisting selfish overlay

applications increase, there is a clear need for strategies to mitigate the inter-overlay interaction. [164] addresses this need by proposing game-theoretic strategies to insure that each streaming overlay network is able to obtain sufficient network bandwidth. There is also past work towards facilitating inter-overlay cooperation in order to allow each overlay to benefit from one another[91]. Our work complements the previous work by investigating the fundamental interaction between the traffic belonging to an overlay application and all other traffic. This interaction is, hence, outside the scope of our work.

2.4 Cross-layer Interaction in BitTorrent Protocol

Sen and Wang[140] perform a systematic characterization of P2P traffic (FastTrack, Gnutella, and DirectConnect) workload observed at a large ISP over a 3 month period. However, their conclusion that P2P traffic can be traffic engineered effectively is not true with BitTorrent traffic where the rate of ingress and egress traffic are positively correlated[47]. Furthermore, our work differs by investigating the impact on TE from the perspective of the stub ASes, where there is a certain level of unfairness in the usage of the multi-homing links. Karagiannis et al.[80] identified that current BitTorrent peer selection is not very ISP-friendly. Using BitTorrent tracker logs and payload packet traces collected at the edge of a 20,000 user access network, they quantify the impact of BitTorrent on end-user experience and resource consumption. They point out that providing locality-awareness decreases the bandwidth usage of the ASes egress link by more than a factor of two and show that the ISP savings increase roughly linearly to the logarithm of active users. Bindal et al.[50] add to previous locality study and propose changing the neighbor selection policy at peers without causing an increase in the peer download latency. However, a main limitation with this strategy is the extra complexity introduced in the neighbor selection, which must be supported by every torrent accessed by peers in an AS. Furthermore, the probability that there are other peers within the same AS is often low, causing this locality-awareness solution to be less effective.

Recently, researchers have proposed strategies to improve cooperation between the P2P applications and the native layer. [1] proposes that ISPs maintain a database that the P2P

users can query to obtain proximity, bandwidth and other network-level information about other peers. This indirect interface can serve to improve the performance of both layers by helping P2P users pick appropriate neighbors and improving the utilization of ISP links. [167] proposes adding an inter-layer interface to enable explicit communication between the P2P applications and the ISP. They present a basic design of the interface (referred to as *iTracker*) and show how adding this improves performance for both layers. Both these approaches fail in the presence of rogue overlays and does not offer sufficient deployment incentive to the ISP. Thus, the deployability of these approaches are suspect.

2.5 Placement of Nodes in Overlays

Although most of our work is concerned with the safe operation of service overlay networks, the design aspects of node placement and neighborhood definition have a huge bearing on the performance achieved. Thus, it is conceivable that an ideal design of an overlay network must take into consideration the forthcoming cross-layer interaction. Numerous studies have investigated the improvement in overlay routing performance achieved by careful placement of overlay nodes and links[142, 95]. The work in [142] performs a gain-cost analysis with the aim of picking the least number of servers and achieving the required gain. Two other efforts on overlay topology design focus on specific routing objectives – J. Han et al.[69] propose ways to aid the robustness of the overlay network, and H. Zhang et al.[174] propose ways to obtain optimal routing cost and utilization. However, the past research work does not directly consider native layer policies and resource limitations, which we specifically address in this thesis. We propose solutions that help determine how the overlay topology can be augmented to retain the routing advantage without causing serious conflict with the native network. Thus, we argue that any overlay node placement strategy should take into account the exact native layer restrictions and enforce them at the overlay layer during the overlay network design process, besides attempting to achieve its local objective.

CHAPTER III

INTERACTION BETWEEN FAILURE RECOVERY IN THE NATIVE AND OVERLAY LAYERS

3.1 Introduction

Typically, in overlay networks, a dynamic routing protocol is employed to adapt overlay routing tables to changing network conditions. At the same time, the native IP network over which the overlay is built also runs its own set of dynamic routing protocols. We are interested in investigating the behavior of this *mixed routing* environment and in particular the characteristics of the interaction between these two routing layers. The complexity in this interaction between routing layers makes it quite challenging to study in general.

To obtain first set of insights into this issue, we focus on the specific problem of *rerouting around failed native links*. An analysis of routing updates on the Sprint network has identified a significant portion (70%) of unplanned network outages to be due to single link failures[105]. It also highlighted that the failures are fairly common even in the modern Internet. This makes the specific problem we are addressing interesting in its own right, in addition to providing insights into mixed routing interactions in general.

We explore the following two scenarios:

- The overlay network is built on top of a single Internet domain and both the overlay network and the native network are running a link-state routing protocol (e.g., OSPF[111])
- Each node in the overlay network resides in a separate Internet Autonomous System (AS). The overlay network runs a link-state routing protocol while a path-vector protocol (e.g., BGP[127]) is run among the ASes.

Before proceeding, we first observe that the mixed dynamic routing environment can be avoided by not using dynamic routing in the overlay layer and always counting on the

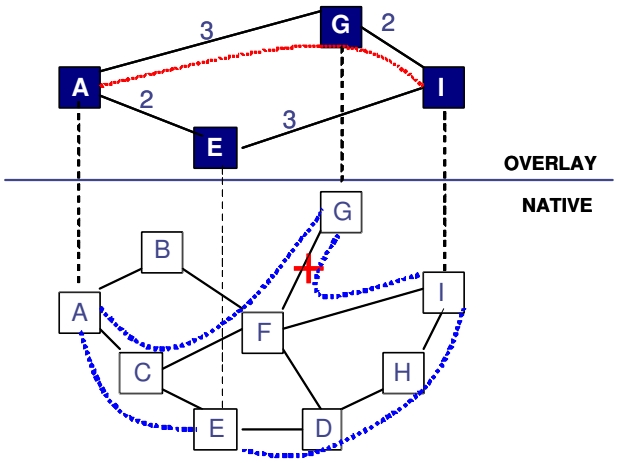


Figure 2: An illustration of an overlay network. The dotted lines in each layer represent the path between the two nodes.

native network to re-configure routes as network conditions change. This is undesirable. To illustrate this, consider, for example, the topology in Fig. 2, where the overlay path AI is statically routed through the overlay node G. When the native link FG fails, the overlay links AG and GI fail. Node G has been separated from the native network. Hence, native rerouting of individual native paths AG and GI will not work. Because the overlay network's routes are static, the overlay network cannot recover from this failure. If on the other hand, the overlay network is capable of performing dynamic routing, then the routing tables may change to allow for the path AEI to be used for overlay traffic. This shows that overlay dynamic routing can significantly enhance the overlay network's survivability.

In our investigation, we first consider a *Dual Rerouting* scenario in which the two routing layers run completely independently. Our goal is to understand the effect of the various settings of protocol parameters on the routing performance as measured by several metrics (discussed in Section 3.2.3. We provide an understanding of Dual Rerouting, its effects and compare rerouting at the two layers. We show that Dual Rerouting provides relatively fast path recovery, although our investigation reveals that Dual Rerouting suffers a performance degradation as a result of the 1) overlap of functionality between the two layers, 2) unawareness of the other layer's decisions, and 3) lack of flexibility. We mitigate these problems by adjusting the overlay layer functioning without affecting the native layer.

Our approach is motivated by the fact that the native network is tuned for the native applications and it seems pragmatic to not alter its operation. In this chapter, we investigate simple schemes that increase the awareness of the native routing protocol and its parameters at the overlay layer. Each scheme provides better control at the overlay through layer-awareness. Specifically, we develop the following three schemes: *Probabilistically Suppressed Overlay Rerouting*, *Deferred Overlay Rerouting* and *Follow-on Suppressed Overlay Rerouting*. We show that with such schemes one can trade off a deterioration in one metric with improvements in another. Though one can devise more complex approaches, it is questionable whether these schemes will be practical. Hence, we restrict our analysis to modest alterations of overlay routing.

The remainder of this chapter is organized as follows. Related work is briefly described in Section 3.6. We explain the model adopted for analysis in Section 3.2. We elaborate on the characteristics of Dual Rerouting in Section 3.3. We develop the layer aware rerouting algorithms in Section 3.4. Simulation results and the corresponding inferences are presented in Section 3.5. We summarize this chapter in Section 3.7.

3.2 Rerouting Model

3.2.1 Framework

Overlay networks represent a virtual network of selected nodes communicating at a layer higher than the network layer. Each overlay node has a well-defined set of neighbors, not necessarily adjacent in the native network. The virtual link between any two overlay nodes constitutes *the native path* between the two nodes. The data communication between the overlay nodes can be accomplished by means of IP-in-IP tunneling[51, 143], where the non-overlay nodes route packets over native links based on the first IP header and the overlay nodes route packets over overlay links based on the second IP header[1]. To facilitate that, the overlay layer maintains a routing table that is independent of the native layer's routing table. We assume a dynamic routing protocol is used to maintain the overlay routing table.

[1]Another approach would use a different non-IP addressing scheme for overlay nodes and an overlay header encapsulated within the native IP headers. The overlay routing table in this case would be indexed by overlay addresses.

In order to get a first set of insights into the issue of inter-layer interaction, we focus on the specific problem of rerouting around failed native links owing to the following two reasons:

- Failures exacerbate any negative effects of the interaction.
- Native links represent the smallest unit of abstraction one can analyze (i.e., failure of any intermediate native node can be translated to the failure of multiple native links)

Past research has primarily dealt with network oscillations caused by load constraints and traffic engineering[83, 98]. Owing to lack of data on actual traffic demands and lack of support for QoS in the current Internet, we do not consider the issue of load in our analysis. Thus, we are more concerned with the change in routes and the corresponding timing, which is a more practical approach applicable under any load condition.

We consider two scenarios:

1. Single-Domain Overlay over Single-Domain Native: In this scenario an overlay network is built on top of a single native network domain. Both networks use a link-state routing protocol (e.g., OSPF). Any of the links in the native network can fail in this scenario.
2. Single-Domain Overlay over Multiple-Domain Native: Here each node in the overlay network resides in a separate Internet AS. The overlay network operates as a single domain and uses a link-state dynamic routing protocol. A path-vector (BGP-like) protocol is used among the ASes to dynamically route around failures in the inter-domain links. To focus on the inter-domain aspects of dynamic routing, we consider only inter-domain link failures in this model. We assume that a link-state routing protocol is used as the intra-domain routing protocol. However, in this scenario, we are not concerned with intra-domain link failures.

The routing protocols adopted in each layer have the following generic parameters:

- An appropriate cost assigned to individual native or overlay links.

- An algorithm to determine the shortest path between two given nodes.

- *KeepAlive messages*: These are exchanged between nodes at both ends of the same native or virtual link for the purposes of link management. The *keepAlive* message exchanges are only of interest when they occur across links that are subject to failure. For example in the single-domain overlay over multiple-domain native case, only *keepAlive* message exchange across overlay links and inter-domain native links are of interest to us.

- A *keepAlive-time*[2]: This represents the frequency at which neighbors exchange *keepAlive* messages.

- A *hold-time*[3]: A native or virtual link is generally assumed to be down if no *keepAlive* messages are received during the hold-time. This parameter directly influences the delay in detecting a link failure.

3.2.2 Rerouting schemes

We consider three approaches for the operation of the dynamic routing protocols at the native and overlay layers:

a) No awareness: This corresponds to what we have dubbed *Dual Rerouting*, where the two layers operate independently to reroute around a failure. This serves as the benchmark for performance comparison.

b) Awareness of lower layer's existence: The overlay layer is aware that the native layer might attempt rerouting on its own, leading to a functionality overlap. This scheme tries to mitigate the problems arising from such an overlap without any further knowledge. We propose two solutions in this context - *Probabilistically Suppressed Overlay Rerouting* and *Deferred Overlay Rerouting*.

c) Awareness of lower layer's parameters: The overlay layer is informed of at least some of

[2]KeepAlive-time is also called hello-interval in certain routers.
[3]Hold-time is also called dead-interval in certain routers.

the native layer routing protocol parameters. We propose one solution in this context - *Follow-on Suppressed Overlay Rerouting*.

We are interested in exposing the performance limitations of the Dual Rerouting approach and then understanding the extent to which functionality overlap can be reduced by adding more awareness of the native layer at the overlay layer. We are also interested in the effect that such increased awareness has on the performance of the mixed routing scheme.

3.2.3 Performance metrics

We evaluate the performance of the mixed routing approaches from the perspective of the availability of the overlay path(s) affected by an individual native link failure. We define *recovery* as the rerouting immediately following a failure. Though a rerouting event at a particular layer does not necessarily cause the traffic to change course, a recovery event does. Our focus is on the performance of the recovery process required to re-establish an overlay path affected by the failure[4]. Such a path will be established when either:

- The native network establishes a new native path between the two ends of the broken overlay link, or
- The overlay network determines a new overlay route between the two end points of the broken overlay path.

The exact order in which these two events occur can affect the routing performance.

We are also interested in evaluating the performance of the routing protocol when the failed native link is repaired.

We use the following four metrics.

3.2.3.1 Hit-time

Hit-time is defined as the time period after a native link failure during which there is no communication between the two overlay nodes at the ends of a failed overlay link. Based

[4]Following standard terminology, an overlay path is made of one or more overlay (virtual) links, which in turn comprises one or more native links.

on the traffic characteristics, the systems incurs a certain amount of packet loss during this period. Hit-time is made up of the following components:

- *Detection time*: This is the time from when a link fails and a node that is at one end of the failed link determines that the link has failed. The failure can happen anytime during the life of the hold-timer. Hence, the detection time is generally a random time between (0, hold-time).
- *Convergence time*: This is the time from when a link fails until all nodes in the network are aware of the failure and the routing changes are enforced[122, 92].
- Time to compute the route
- Device time to activate the new route

When both layers attempt to reroute around a failure, hit-time is the time taken by the layer that completes rerouting first. Thus, we define the *effective detection time* as the smaller of the detection times at either layer. If neither layer is able to reroute around the native link failure, the hit-time is infinite.

3.2.3.2 Success rate of rerouting

The success rate of rerouting is defined as the proportion of failed overlay paths that get rerouted successfully. A path may not be rerouted successfully because a native link failure caused a partition in the native network or the overlay network.

It can be inferred from the cumulative distribution of hit-times observed for each affected overlay path.

3.2.3.3 Number of route flaps

During a rerouting process, the route used by the overlay link is changed to avoid the failed link. This can happen multiple times and can be the result of the overlay or the native dynamic routing process. Each such change is defined as a *route flap*[5]. Route flaps can

[5]Also called route oscillation.

be a serious problem in case of TCP and other traffic that relies on packet ordering[17]. It not only increases the packet loss and latency, but also burdens the network with extra path computation overhead. As a result, the network efficiency will be degraded and end-to-end performance is hurt. The route flaps are also an indication of the instability in the system. However, these route flaps are not persistent (i.e., overlay routing converges eventually[98]) and may not impact the traffic any greater than the actual loss of packets. Nevertheless, they trigger oscillations which take a longer time to resolve in the presence of load constraints or traffic engineering.

For our analysis, we assume that all failed native links are repaired at some time instance. Such link repair is first detected by the native layer and then communicated to the overlay layer by means of updated overlay link cost. This repair may also cause another route flap as the network switches back to the original (pre-failure) route.

Each path affected by a native failure can display a different sequence of route flaps, based on the temporal dynamics and the layer yielding the optimal path. Hence, we are interested in the average number of route flaps, calculated as:

$$\text{Average route flaps} = \frac{\text{Number of route flaps}}{\text{Number of failed overlay paths}}$$

In our scenarios, one of the two layers definitely performs rerouting, unless a failure causes the network to be disconnected. A maximum of one route flap occurs following a repair and it depends on the state of the system at the end of the previous rerouting operation.

3.2.3.4 Path cost inflation due to a route flap

Path cost inflation represents the increase in the cost of the path between the overlay nodes at the ends of the overlay link(s) broken as a result of a native link failure. We measure this cost relative to the cost of the original (pre-failure) path used by the overlay link. It conveys the penalties for choosing a longer path. The path cost inflation ratio is defined as:

$$\text{Path cost inflation} = \frac{\text{Path cost after rerouting}}{\text{Path cost before failure}}$$

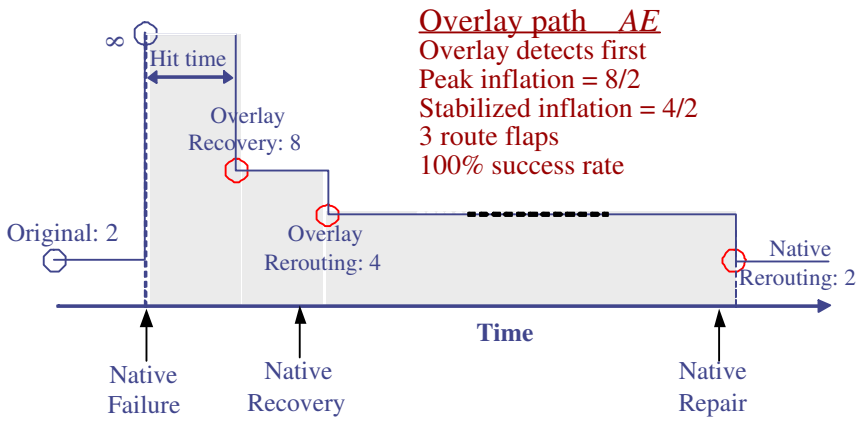


Figure 3: Sequence of route flaps and the path cost progression for link failure in the example. The numbers indicate path cost in terms of native hop count. In the above figure, the overlay layer detects failure first.

We are interested in this metric after a sufficient time has elapsed since native link failure but before native link repair. This value shall be referred to as *stabilized inflation*. The dynamic protocols in both layers constantly strive to achieve the least path cost. Hence, after a sufficient time has elapsed the path cost inflation of all affected paths will be the same across different schemes.

Immediately after a failure, path cost inflation starts at ∞ when the ends of the path are disconnected. Once a path is re-established as a result of the rerouting process, a path cost inflation attains a finite value. This inflation may decrease over time as the rerouting process continues. Thus, we are also interested in the maximum (non-infinite) path cost inflation observed per failed overlay path as a result of rerouting. This value, referred to as *peak inflation*, indicates the level of sub-optimality suffered by the overlay path before stabilizing. When the stabilized inflation is constant across different schemes, the peak inflation acts as a measure of the overall sub-optimality.

We use the notations T_{peak} and T_{stable} to indicate the time instance when the peak inflation and stabilized inflation are observed respectively. T_{stable} represents the time instance when the system attains steady state and should be as small as possible.

We illustrate the dynamics of the performance metrics in Fig. 3 where the numbers indicate the actual overlay path cost, in terms of native layer hop count. Consider the

example topology in Fig. 2 where both layers use hop count as the link cost. When the native link CE fails, the overlay link AE is broken. Assume the overlay layer detects the failure first. The link is now overlay-rerouted through G and I leading to one route flap and a sub-optimal path of cost 8. Once the native timers expire, the native path AE is native-rerouted through F and D. The new native path is now the optimal one with a cost of 4. The overlay senses this and adopts the direct link, thereby causing another route flap. The system is now stable until the native link is repaired. Once the native repair is triggered, the native path switches back to the original path as it provides a lower cost of 3. This leads to a third route flap. Thus we get the performance values mentioned in Fig. 3.

3.3 Characteristics of Dual Rerouting

Dual Rerouting refers to the scheme where each layer operates completely independent of the other. This independent operation leads to a large number of route flaps that cause wastage of resources[6], affect the performance of traffic that needs accurate packet ordering and lead to general system instability. The overlay path cost in-between the route flaps can be sub-optimal as the overlay network is unaware as to which layer provides the optimal rerouting.

Assuming we have no control over the native layer parameters, these problems in Dual Rerouting can be mitigated by adjusting the values of hold-time, keepAlive-time and cost scheme at the overlay layer. By tuning the timers at the overlay layer, we vary the layer at which failure detection is likely to happen first. We also investigate waiting for fewer *keepAlive* messages without risking any false alarms in failure detection. Section 3.5 presents the effects in detail.

Past research on resilience suggests adopting a low keepAlive-time at the overlay layer so that it can detect the failure first[6]. This tends to aggravate the problems in Dual Rerouting by instigating both layers to perform rerouting. We investigate the associated negative effects and determine whether an earlier overlay detection is completely beneficial.

[6]Time for computing new routes and bandwidth overhead for exchanging routing protocol updates.

Notice that Dual Rerouting has the best hit-time compared to any other scheme as there are two different layers aiming at rerouting around the failed link. Hence, it is not possible to improve the hit-time performance any further. Dual Rerouting, however, performs poorly with respect to the other metrics (viz. number of route flaps and path cost inflation). We, therefore, consider layer-aware schemes that allow us to trade off improvements in these other metrics with longer hit-times.

3.4 Layer Aware Overlay Rerouting

In this section, we present three native layer aware overlay rerouting schemes - *Probabilistically Suppressed Overlay Rerouting*, *Deferred Overlay Rerouting* and *Follow-on Suppressed Overlay Rerouting*. These schemes require knowledge of the native layer's routing protocol existence and in the case of the last scheme some minimal knowledge of the state of the native layer rerouting efforts.

The layer-aware schemes operate based on the assumption that the native layer cannot be modified. This is because the overlay networks share the native network with other non-overlay applications. The operation of the native network is thus tuned for the native applications. We are, therefore, only allowed to control the overlay layer by suppressing or delaying the rerouting process. It is possible to construct more complex inter-layer coordination and information exchange (e.g., assuming that the overlay layer has knowledge of the native layer topology[69, 27] or of the other coexisting overlay networks in the system[91]). However, this may not be very practical. More importantly, we find that significant control over the tradeoffs between hit-time and the other metrics is possible through the simple approaches we consider.

Probabilistically Suppressed Overlay Rerouting In this scheme, we suppress overlay rerouting without any particular knowledge of the native layer. The suppression operation is done with probability p on each overlay rerouting attempt (irrespective of whether it follows a failure or a repair). Various values of p lead to different recovery performance and are of interest to us. The main advantage to be gained with the suppression operation is the decrease in the number of route flaps. If the value of p is configured appropriately for an

overlay path, it is possible to achieve the best path cost values in between the route flaps. Profiling the performance for each p will help in choosing the right value for the network in consideration. A value of 0 for p corresponds to the case of Dual Rerouting. When p is 1, all overlay rerouting attempts are suppressed and there is a possibility the overlay path may never be rerouted if the native rerouting does not succeed.

Deferred Overlay Rerouting In this scheme, we delay overlay recovery by a constant value to give the native layer a chance to recover the path. After that time has elapsed, if the native network has not yet recovered, overlay recovery is performed. As the overlay network is made to wait until the native layer rerouting, the system incurs a longer hit-time. The duration by which we delay overlay recovery is determined based on the amount of additional hit-time the overlay traffic can handle. This scheme will have fewer route flaps relative to Dual Rerouting and a longer hit-time.

Follow-on Suppressed Overlay Rerouting In this scheme, the overlay layer keeps track of the native layer's timer values. As the overlay layer hold-timers are independent from that of the native layer's, there is a possibility the native layer keepAlive-timer closely follows that of the overlay layer, relative to the time of failure. The overlay rerouting effort might be wasteful because the native layer recovery is not too far off. The decision of whether to suppress the overlay rerouting depends on the *follow-on time*, which is defined as the time remaining for the next native layer detection. Fig. 4 illustrates the follow-on time which must be known at the overlay layer to avoid undesired recovery and functionality overlap between layers. If the follow-on time is less than a particular threshold value, the overlay rerouting is suppressed. The concept of follow-on time is applicable only when the overlay layer detects the failure first. The threshold value needs to be determined by the amount of additional hit-time the system can tolerate.

The characteristics of this scheme are similar to *Deferred Overlay Rerouting*, except that this scheme has a relatively smaller hit-time. This is because the overlay rerouting is initiated right away in cases where the follow-on time is higher than the threshold, unlike *Deferred Overlay Rerouting* where the overlay layer always waits for the constant delay

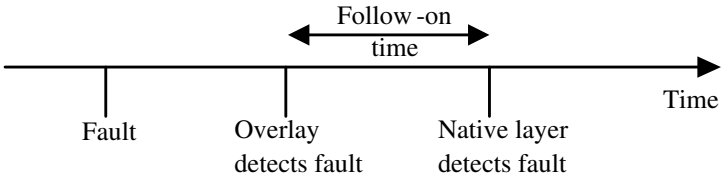


Figure 4: Possible scenario for failure detection in two layers. The decision in the *Follow-on Suppressed* scheme depends on the amount of time between the overlay detection and the native layer detection.

period anyways. This scheme establishes an upper bound on the hit-time observed.

This scheme requires the knowledge of when the timer expires at the node closely associated with the failure. Obtaining this information is even harder in the multi-domain scenario. We postulate that signaling between layers, in combination with explicit BGP peering, can accomplish this, but the details are outside the scope of this thesis. We are interested first in measuring the benefits we derive from this scheme before developing techniques to do it.

3.5 Results

We now present NS-2 simulation results for Dual Rerouting and the three layer-aware schemes proposed. The strategies work similarly in single-domain and multi-domain scenarios. Hence, the results are presented together and the differences mentioned when applicable.

3.5.1 Simulation Setup

It should be noted that the performance of the rerouting schemes is topology-specific. Hence, we need to simulate multiple overlay topologies over multiple native networks to improve the generality of the results. We use GT-ITM[23] to generate random network topologies for the simulations. We generate 5 native network topologies and 5 overlay topologies at random. The 25 possible combinations generated by mapping the overlay topology to the native topology are simulated and the results averaged over them. The links used in the simulation are bi-directional and each end triggers the detection process once a native link fails. We assume that the native and overlay layers perform symmetric routing. This is reasonable as the average statistics will still be the same.

The intra-domain native paths use hop-count based costs (by setting a native link's cost to 1), while the inter-domain native paths use the length of the AS path. The overlay paths use a native hop count based cost scheme[7].

The next heavily influencing parameter is the hold-time of each layer. To insure consistency of treatment, all solutions use the same timer values at the native layer. We assume here that the timer values are consistent between both ends of any link. The lowest configurable keepAlive-time for commercial routers (such as the Cisco 7200 series router) is 1 sec. We adopt the same value for the intra-domain keepAlive-time. The native layer waits for the absence of three consecutive *keepAlive* messages before declaring a link failure. Hence, the native hold-time is 3 secs. Typically, the time-scale of operation for IGP and EGP fault detection differ substantially and are often incomparable. For inter-domain native links, we set the keepAlive-time and hold-time as 5 secs and 15 secs respectively. The native hold-times are close to the practical values used by modern routers[97]. At the instance of a failure, the native detection time is randomly calculated in the range of (hold-time - keepAlive-time, hold-time) and detection is enforced at its expiry.

In the overlay layer, we have complete freedom in configuring the keepAlive-time and the hold-time. As hold-time represents the time period during which no *keepAlive* messages are received, it is a multiple of the keepAlive-time period. In this chapter, we consider hold-time being two times the keepAlive-time and three times the keepAlive-time. Waiting for too few *keepAlive* messages can cause the node to mistake congestion effects as failure.

3.5.1.1 Network Topology

For the single-domain native network scenario, each native topology we simulate contains 100 nodes, while each overlay topology contains 10 nodes. The placement of overlay nodes and link connectivity at either layer are decided at random.

For the multi-domain native network scenario, each native topology we simulate contains 500 nodes, while each overlay topology contains 10 nodes. There are 20 stub domains with 24 nodes each and 5 transit domains with 4 nodes each. The connectivity is decided at

[7]We did not see much difference when experimented with an overlay hop-count based cost scheme.

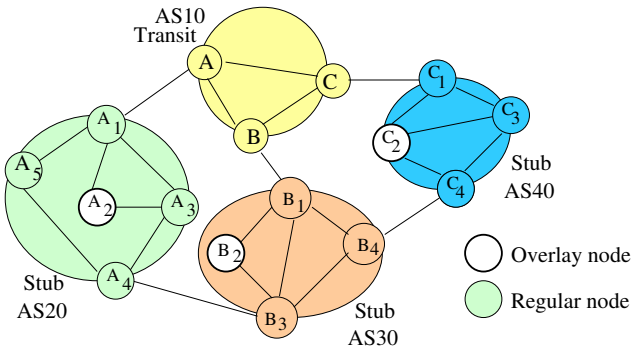


Figure 5: Advantages in overlay rerouting in the multi-domain scenario. The double-circled nodes represent the overlay node

random. The non-border node with highest edge degree is selected as the overlay node in each stub-domain. Thus, no two overlay nodes are in the same domain. The stub networks that host overlay nodes are multi-homed to incorporate redundancy and increase the success rate of rerouting.

We experimented with multiple choices for topology size in both scenarios and got similar results. This implies that the effect of topology size is negligible. Hence, we only present the results for the above mentioned choice.

BGP++ is used for simulating the inter-domain routing dynamics[36][8]. We use the *community* configuration in the simulator to enforce the policy that the private stub-stub links are to be used only for exchanging native traffic between the two stubs. This policy brings in an added advantage for overlay rerouting which will be able to use a redundant stub-stub link. Consider the topology in Fig. 5 where the stub-stub link A_4B_3 can be used only to exchange traffic between the two domains - AS20 and AS30. Initially, the overlay path A_2C_2 passes through AS10. In case of failure in link AA_1, the native network will be unable to reroute the overlay link. This is because it cannot use the private link A_4B_3 due to policy constraints. However, the overlay layer will be able to reroute the traffic over the two overlay links A_2B_2 and B_2C_2. In certain topologies, the overlay-rerouted path can be optimal because it does not have to pass through the transit domain AS10.

[8]The overlay nodes in our simulation do not try to explicitly peer with any BGP router and are hence oblivious to the inter-domain dynamics.

3.5.1.2 Link Failure Modeling

Failure modeling is complicated as failure history information is unavailable for the Internet or other overlay networks. To avoid a specific failure model and its set of assumptions, we use a *stateless all-link failure* approach. In this approach, all of the native links (intra-domain links in the case of single-domain and inter-domain links in the case of multi-domain) are failed one at a time and the statistics tabulated for all possible overlay paths. The state of the system is reset before simulating each failure. This helps study the influence of failure in any location of the network and tries to reduce the dependence on the topology. Occasionally, both overlay rerouting and the native rerouting fail. This can happen when the failure of a particular link breaks the native network into more than one component, or when the policy restrictions prevent the native network from using the available gateway routers. Our simulation results do not include such cases.

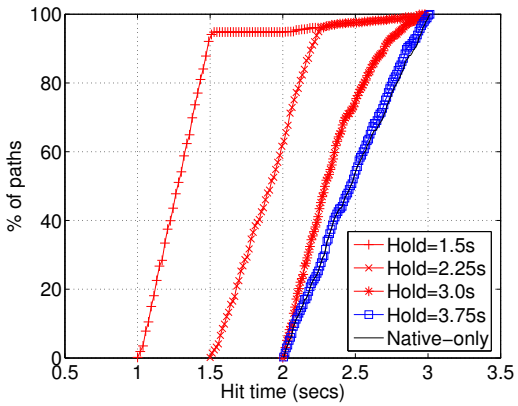
3.5.2 Dual Rerouting

We simulated Dual Rerouting and measured the performance of overlay rerouting based on hit-time, number of route flaps, and path cost inflation. This section presents those results and derives its relation to the hold-time and keepAlive-time.

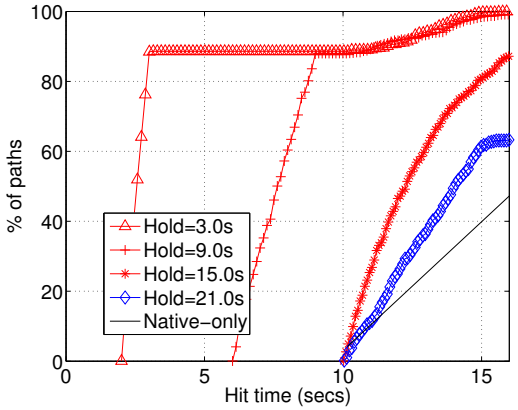
Note that the values we present in the tables are normalized against those observed for native-only rerouting, unless mentioned otherwise. We expressed these normalized values in terms of percentage.

Fig. 6 shows the cumulative distribution function (CDF) of hit-times experienced by overlay paths in our simulations. Recall that hit-time is made up of two main components - the detection time, which depends on the hold-time and the convergence time, which depends on the dynamics of the routing protocol in consideration[111, 92]. Fig. 6(a) and Fig. 6(b) show that increasing the overlay hold-time gradually increases the observed hit-time until the overlay hold-time is equal to the native hold-time. When this happens, the native layer completes the rerouting first and further increases in overlay hold-time have no effect. Hence, the curves begin to merge for higher values of overlay hold-time.

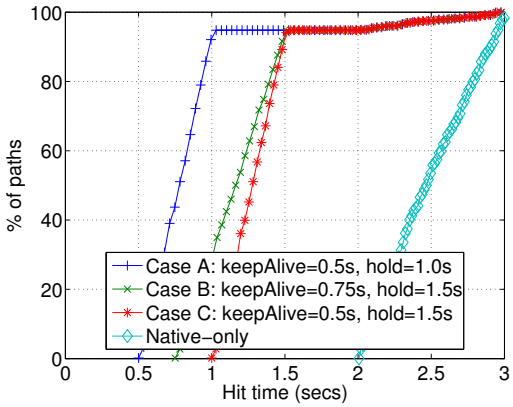
The overlay hold-time was varied between twice the keepAlive-time (case A, B) and



(a) Single-domain
(hold-time=3*keepAlive-time)



(b) Multi-domain
(hold-time=3*keepAlive-time)



(c) Single-domain
(Comparing keepAlive schemes)

Figure 6: Cumulative distribution of hit-time for different overlay timer values, when path cost is native-hops

Table 2: Average number of route flaps

—**Single-domain**—

	Hold=1.5s	**Hold=3.0s**	**Hold=4.5s**	**Native-only**
% overlay recovered	95.8	45.0	0	0
% native recovered	4.2	55.0	100	100
Average route flaps	140.58%	125.08%	109.57%	1.567
	(Normalized by native-only)			(Absolute)

—**Multi-domain**—

	Hold=9s	**Hold=15s**	**Hold=21s**	**Native-only**
% overlay recovered	94.2	52.2	21.4	0
% native recovered	5.8	47.8	78.6	78.8
Average route flaps	157.25%	153.60%	150.53%	1.207
	(Normalized by native-only)			(Absolute)

three times the keepAlive-time (case C). Fig. 6(c) shows that case A has the lowest hit-time. When the keepAlive-time is the same, case A has a smaller hold-time. When the hold-time is the same at 9 secs, case B has a bigger detection window of (4.5s, 9s), unlike case C that has a narrower detection window of (6s, 9s). These two factors cause the earlier detection when hold-time is twice the keepAlive-time.

The knee bend in the graphs of Fig. 6 corresponds to the time instance when the overlay layer failure detection is complete. The flat section of each curve corresponds to the idle time where the system waits to perform native rerouting on the unrecovered overlay links.

The average number of route flaps per link failure has been listed in Table 2. We also present the percentage of failed overlay paths successfully recovered at a particular layer. The sum of the two values (*% overlay recovered* and *% native recovered*) reflect the success rate of the rerouting scheme. From the table, we can observe that the overlay layer can recover a maximum of 95.8% of the paths in the single-domain scenario and 94.2% in the multi-domain scenario. We also observe that native-only rerouting has a success rate of 100% in the single-domain scenario, in contrast to a 78.8% success rate in the multi-domain scenario. This demonstrates the importance of dynamic overlay rerouting.

We also infer from Table 2 and Fig. 6 that the overlay hold-time is inversely proportional to the number of route flaps, while being directly proportional to the hit-time. This shows a clear tradeoff between the number of route flaps and hit-time.

Table 3 shows the average values of path cost inflation for different overlay hold-times.

We see from the table that the hold-time does not have a direct influence on the individual path cost inflation. By regulating the proportion of overlay rerouting and native rerouting, hold-time controls the extent of inflation. Hence, the value of peak inflation decreases with an increase in hold-time. We also see from the peak inflation value that native-only rerouting is the best, though it tends to have a low success rate in the multi-domain scenario. The stabilized inflation for different hold-times are almost equal, implying that Dual Rerouting attains the same inflation in steady state.

In the single-domain scenario, the overlay paths at the end of native link repair were noticed to be of the same length as the original path. But, this is not true in the case of multi-domain scenario where the final path cost was different than the original one. This is because the native network computes paths using the AS path length and not hop count. Two AS paths of equal length can have different number of hops as it does not publicize the internal routes. Hence, the cost ratio does not necessarily fall back to 1 after link repair.

Table 3 also presents results for the time instance of peak inflation and stabilized inflation. The table shows that, for both single-domain and multi-domain scenarios, the T_{peak} is closely related to the overlay hold-time. This confirms that overlay rerouting is what causes the peak inflation. In single-domain scenarios, the T_{stable} is reduced when the overlay hold-time exceeds native hold-time because the native layer rerouting typically occurs first and provides the stabilized inflation. But, this is not observed in multi-domain scenarios as a certain proportion of overlay paths achieve their least cost route with overlay rerouting.

We also computed the 95% confidence interval for each of the statistics presented in this chapter. We noticed that the confidence interval stretches to a maximum of $\pm 4\%$ of the mean value.

Summary of factors affecting Dual Rerouting:

From the above simulations, we observe that native rerouting provides the optimal alternate path in both the single-domain and multi-domain scenarios, though it suffers from a low success rate in the latter case. Thus, we can obtain lower number of route flaps and lower peak inflation by giving a higher precedence to the native rerouting attempts. We

Table 3: Average path cost inflation

—**Single-domain**—

Hold-time	**1.5s**	**3.0s**	**4.5s**	**Native-only**
	(Normalized by native-only)			(Absolute)
Stabilized inflation	Equal value: 100%			1.202
$\mathbf{T}_{stable}$ **(secs)**	97.98%	113.70%	107.25%	2.481
Peak inflation	130.28%	114.22%	100%	1.202
$\mathbf{T}_{peak}$ **(secs)**	53.44%	94.23%	100%	2.481

—**Multi-domain**—

Hold-time	**9s**	**15s**	**21s**	**Native-only**
	(Normalized by native-only)			(Absolute)
Stabilized inflation	Equal value: 108.59%			1.117
$\mathbf{T}_{stable}$ **(secs)**	90.81%	114.66%	125.46%	12.41
Peak inflation	134.29%	117.81%	106.71%	1.117
$\mathbf{T}_{peak}$ **(secs)**	67.22%	98.39%	108.78%	12.41

adopt this form of control as the fundamental strategy behind the design of the layer-aware schemes, where the tradeoffs between the performance metrics can be controlled by the three parameters - *probability of suppression*, *delay* and *follow-on threshold*.

Typically, the overlay timers are calculated based on the desired amount of routing protocol overhead and packet loss during failure. Performance of Dual Rerouting can be substantially improved by adjusting just the value of overlay hold-time, which has the same effect as suppressing the overlay rerouting operation. The following choices are recommended for achieving optimal performance with Dual Rerouting:

- Overlay hold-time value very close to the native hold-time, irrespective of the keepAlive-time chosen.
- Declare failure after the absence of two *keepAlive* messages, rather than three.

In the rest of the section we discuss layer-aware rerouting schemes. They use the above observation of Dual Rerouting as the base line to enhance its performance. Hence, the keepAlive-time for native and overlay were set to the same value (1 sec in the single-domain scenario and 5 secs in the multi-domain scenario). This implies that native rerouting can lag overlay rerouting only by a maximum value equal to the keepAlive-time. The hold-time for both layers were set to three times the keepAlive-time of that layer.

Table 4: Performance analysis of Probabilistically Suppressed Overlay Rerouting

Probability of suppression	**0.25** (Normalized by native-only)	**0.5** (Normalized by native-only)	**0.75** (Normalized by native-only)	**Native-only 1.0** (Absolute)
—Single-domain—				
Avg. hit-time for recovered paths	95.60%	96.89%	98.50%	2.481 secs
Stabilized inflation	103.24%	104.24%	102.99%	1.202
Peak inflation	112.48%	110.00%	109.23%	1.202
—Multi-domain—				
Avg. hit-time for recovered paths	98.94%	99.18%	100%	12.405 secs
Success rate	92.8%	84.1%	79.5%	78.8%
Stabilized inflation	111.19%	108.68%	105.37%	1.117
Peak inflation	115.04%	110.74%	105.82%	1.117

3.5.3 Probabilistically Suppressed Overlay Rerouting

The main parameter we can control in this scheme is the value of p. Simulation results, obtained by suppressing all the overlay rerouting operations with a constant probability, have been tabulated in Table 4. The table shows that the hit-time increases on increasing the probability p for suppressing. This scheme suffers a higher hit-time than Dual Rerouting because some of the earlier overlay recovery attempts are disabled and the failed paths are made to wait longer. The hit-time is observed to be in-between native-only and Dual Rerouting.

As the suppression probability increases, some of the overlay rerouting operations required to achieve the optimal path may be suppressed. Hence, the stabilized inflation tends to decrease, as can be seen from the table. Table 4 also shows that the peak inflation decreases gradually with an increase in p because more native rerouting operations, that yield shorter paths, are performed. In the multi-domain scenario, we see that the success rate decreases almost linearly with an increase in p. This is because the native layer was unable to recover the path after overlay rerouting was completely suppressed. We do not present the values for the number of route flaps as the results are straightforward - increasing the probability of suppression causes a decrease in the number of route flaps. The above mentioned trends represent the tradeoffs that can be used to select the value of p.

Table 5: Performance analysis of Deferred Overlay Rerouting

—**Single-domain**—

Delay	**0.250s**	**0.375s**	**0.5s**	**Native-only**
	(Normalized by native-only)			(Absolute)
Avg. hit-time for recovered paths	97.70%	98.75%	99.43%	2.481
Peak inflation	111.89%	110.73%	108.48%	1.202

—**Multi-domain**—

Delay	**1.5s**	**2.0s**	**2.5s**	**Native-only**
	(Normalized by native-only)			(Absolute)
Avg. hit-time for recovered paths	104.24%	106.05%	107.75%	12.405
Peak inflation	115.66%	114.32%	113.60%	1.117

3.5.4 Deferred Overlay Rerouting and Follow-on Suppressed Overlay Rerouting

The *Follow-on Suppressed Overlay Rerouting* tends to reroute paths in the overlay layer right away, if the follow-on time is higher than the threshold. Thus, *Follow-on Suppressed Overlay Rerouting* has a smaller hit-time compared to *Deferred Overlay Rerouting*. Intuitively, the hit-time increases with an increase in delay or follow-on threshold because the network has to wait for the native recovery to be initiated. The hit-times of both these schemes are higher than in Dual Rerouting as some of the overlay recovery operations are suppressed or postponed. As no overlay rerouting is suppressed indefinitely, these schemes have a 100% success rate and obtain the optimal path cost eventually.

Both schemes have similar characteristics in terms of route flaps and path cost inflation, as they perform exactly the same sequence of overlay rerouting operations (albeit at a slightly different time). The performance of the *Deferred Overlay Rerouting* scheme for different delay values is tabulated in Table 5. From the table, it can be seen that a higher delay implies a reduced number of overlay rerouting attempts and thereby shorter paths. Hence, the peak inflation decreases with an increase in the delay. As with the previous scheme, we do not present the values for the number of route flaps as the results are straightforward - increasing the delay causes a decrease in the number of route flaps.

3.5.5 Performance Comparison

This section compares the performance of Dual Rerouting, *Probabilistically Suppressed Overlay Rerouting*, *Deferred Overlay Rerouting*, and *Follow-on Suppressed Overlay Rerouting*. We set the overlay layer hold-time to the same value as the native layer hold-time. The delay and follow-on threshold were set to 0.375 secs or 2 secs depending on the scenario and the suppression probability was set to 0.5.

Table 6 shows the route flap and path cost inflation statistics for the layer-aware schemes and the native-only rerouting scheme. Among the layer-aware schemes, we observe from the table that *Probabilistically Suppressed Overlay Rerouting* has the lowest route flaps, because it suppresses more overlay rerouting operations. For the same reason, it has lower peak inflation value. *Deferred Overlay Rerouting* does not suppress any overlay rerouting operation after traffic recovery. Hence, it has the lowest stabilized inflation. *Deferred Overlay Rerouting* also has a higher success rate because we will never eliminate overlay rerouting completely.

Based on Table 6, we can not comment on any particular relation between the different values of T_{stable} or T_{peak} because the actual overlay rerouting operations that were suppressed were random. However, the results are consistent with the following two observations. 1) Native-only rerouting must attain steady state much faster than the other schemes. 2) Dual Rerouting, which has no suppressed overlay rerouting operations, must attain the peak earliest.

Fig. 7 shows the CDF of hit-time for the three rerouting schemes. Dual Rerouting is always the best as there are more overlay rerouting operations trying to recover the failed path. This comes at the expense of more route flapping during recovery as shown in Table 6. The curves for *Probabilistically Suppressed Overlay Rerouting* and *Follow-on Suppressed Overlay Rerouting* closely follow each other because they recover an approximately equal proportion of the failed links in the overlay layer. *Deferred Overlay Rerouting* has higher hit-time as it delays the overlay rerouting irrespective of the subsequent native rerouting.

Summary of performance of layer-aware schemes:

None of the three layer-aware schemes are the best according to all of the four performance metrics, but they provide the best hybrid situation. Based on whether the system is sensitive to hit-time, route flap or path cost inflation a different scheme can be chosen. Choosing the right scheme also depends on the amount of awareness the overlay layer has. In certain cases, Dual Rerouting can be made to perform best by varying the hold-time and keepAlive-time.

We notice that *Follow-on Suppressed Overlay Rerouting* does not provide a substantial advantage over *Deferred Overlay Rerouting*. Hence, the increase in implementation complexity is not justified. In most cases, the other two simple layer-aware schemes are capable of providing the required control.

The exact degree of control can be varied by tuning the five parameters - namely keepAlive-time, hold-time, suppression probability, delay and follow-on threshold. Tuning these parameters in a real-life overlay network requires the operator to perform a tradeoff analysis similar to our methodology and make a multi-objective decision for the appropriate traffic type. However, we are unable to posit a general rule of thumb that can be applied widely. This is because the importance of each metric is unequal and is specific to the type of traffic. In most situations, packet loss has a more serious effect on the performance of the overlay traffic and therefore reducing the hit-time should be given a high precedence. Hence, we can conclude that Dual Rerouting, despite its problems with the functionality overlap, can be considered to achieve the most desirable performance. This shows that the practical limitations brought by the lack of information cannot be circumvented. We recommend tuning of the overlay layer timer values, in combination with vanilla Dual Rerouting, as the best rerouting scheme in most systems.

3.6 Related Work

A complementary work on failure detection algorithms for overlay networks can be found in [179]. This work aims at reducing detection time, probability of false positive, control overhead and packet loss rate, by using information sharing across overlay nodes and storing state information of neighbors. Our work is independent of the detection algorithm used and

Table 6: Comparison of all rerouting schemes

Type of rerouting	Dual Rerouting	Suppressed	Deferred/ Follow-on	Native-only
	(Normalized by native-only)			(Absolute)
—Single-domain—				
Average route flaps	125.08%	101.59%	109.85%	1.567
Stabilized inflation	100%	108.32%	100%	1.202
$\mathbf{T}_{stable}$ **(secs)**	113.70%	100.48%	107.33%	2.481
Peak inflation	114.22%	109.98%	110.73%	1.202
$\mathbf{T}_{peak}$ **(secs)**	94.23%	96.89%	98.75%	2.481
—Multi-domain—				
Average route flaps	153.60%	114.00%	146.56%	1.207
Success rate	100%	84.1%	100%	78.8%
Stabilized inflation	108.59%	112.23%	108.59%	1.117
$\mathbf{T}_{stable}$ **(secs)**	114.69%	104.93%	117.42%	12.405
Peak inflation	117.81%	110.74%	114.32%	1.117
$\mathbf{T}_{peak}$ **(secs)**	98.45%	99.18%	106.05%	12.405

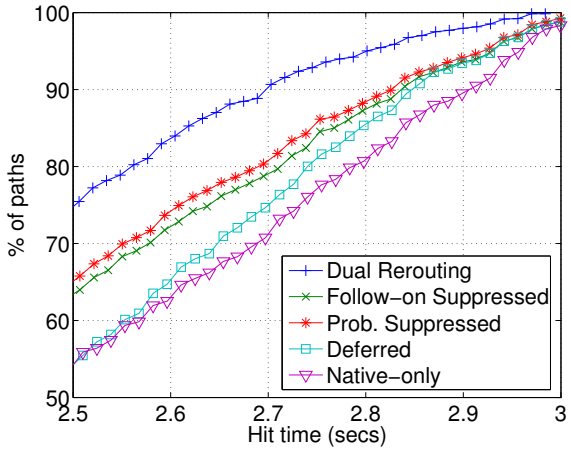


Figure 7: Hit-time comparison of all schemes in the single-domain case

hence orthogonal to their work. The work in [83] emphasizes the importance of a synergistic co-existence between the native and overlay layers, in the light of load constraints. It also characterizes the oscillations that occur due to the presence of multiple independent overlay networks. In our work, we do not consider load constraints as it is not yet a practical feature in most overlay networks. We are trying to address the more fundamental questions of functionality overlap and layer awareness.

Recently, researchers analyzed the interactions between selfish overlay routing and traffic engineering in a game theoretic approach[100, 98]. Unlike our work, the game theory analysis applies only to cases where the native and overlay layers operate independent of each other. The previous work does not consider the interaction of protocol timers and fails to present ways to mitigate the effects of the interaction.

3.7 Summary

In this chapter, we investigate a mixed routing environment in which an overlay network deploys a dynamic overlay routing protocol on top of an existing native network dynamic routing protocol. We focused on the interaction between the two routing layers and their performance when rerouting around native link failures. The Dual Rerouting scheme, in which the layers operate independently, was sub-optimal in terms of the number of route flaps and the overlay path cost inflation, though it was able to provide the fastest path recovery. To reduce the sub-optimalities, layer awareness is crucial. We considered three schemes for intelligent overlay rerouting in the specialized layer-aware overlay network. The *Follow-on Suppressed Overlay Rerouting* and *Deferred Overlay Rerouting* schemes perform the best in terms of path cost inflation and success rate. The *Probabilistically Suppressed Overlay Rerouting* has the least number of route flaps. By using *Follow-on Suppressed Overlay Rerouting* scheme in networks that allow the overlay layer to access the native layer attributes, we can obtain hit-time lower than *Deferred Overlay Rerouting*. While more complex layer-aware rerouting schemes are possible, our work shows that the relatively simple schemes we consider provide us with sufficient flexibility to control the tradeoffs between the various rerouting performance metrics. The improvements achievable using

the scheme mentioned above, however, are limited when we have no control over the native layer. Thus, we envision that as overlay applications proliferate, the native layer should gradually evolve to suit the overlay network requirements. We investigate in Section 7.4 an alternate approach that obtains more benefits through modification of the native layer functioning.

CHAPTER IV

ANALYZING INTER-DOMAIN POLICY VIOLATIONS IN OVERLAY ROUTES

4.1 Introduction

The Internet is a complex structure arising from the interconnection of numerous autonomous systems (AS), each exercising its own administrative policies to reflect the commercial agreements behind the interconnection. However, routing in service overlay networks is quite capable of violating these policies to its advantage. In this chapter, we address a fundamental question with regards to the impact of overlay routing on the enforcement of *native network routing policies*. More specifically, we are interested in the extent to which overlay paths violate AS transit policies and exit policies.

Consider, for example, a hypothetical AS-level connectivity graph as show in Fig. 8. In that figure, nodes A, B and C are overlay nodes trying to obtain the best possible route to each other. Node B can route data to node C using the overlay path BAC, which results in University X's AS being used for transiting traffic between University Y and Commercial organization Z. This is a violation of the AS transit policy at University X. From an economic perspective, we see that University Y saves money paid to Provider 2, by not using the legitimate route between nodes B and C. This saving comes at the expense of University X, which is not part of any transit agreement. Furthermore, the overlay node

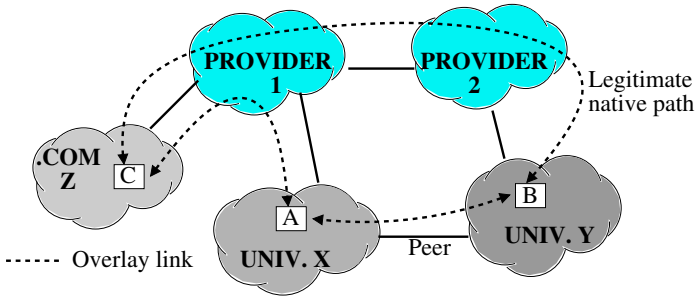


Figure 8: Policy violations using overlay routing

in University Y sends commercial traffic on an academic link, which is a violation of the end-user agreement, thereby representing an exit policy violation at University Y. Because overlays operate at the application layer, both these violations typically go undetected by the native layer.

We start our investigation by evaluating a case study overlay network constructed over the PlanetLab testbed[121] which provides insights into the frequency and characteristics of the different violations (results are reported in Section 4.4). In our dataset, it is interesting to note that close to 70% of the multi-hop overlay routes violate the native layer transit policies and over 87% of the multi-hop overlay routes violate the exit policies. It is worthwhile to observe that these native policy violations are not a serious issue if the overlay traffic is a minor component of the overall traffic. But, in some cases it is already a significant portion of Internet traffic. It is also likely that overlay traffic will experience significant growth in the future.

As awareness of the impact of overlay applications increases, there is an impending drive to incorporate extra complexity at the native layer to manage the overlay traffic [52, 108, 80]. There have been two types of commercial solutions proposed for both enterprise networks and service provider networks:

- Those that help manage overlay traffic without impacting the user experience[80, 21, 134]
- Those that help filter out overlay traffic without concern for the user experience[161, 117, 144]

This second class of solutions motivates us to investigate the impact of their deployment to counter the inter-domain policy violations committed by overlay routes. Specifically, we focus on two types of overlay traffic filtering – *blind filtering* and *policy-aware filtering*. We show that such filtering can be detrimental to the performance of overlay routing.

There exists two forms of overlays that can cause native policy violations - i) service overlays[39], where an overlay service provider (OSP) purchases resources from the underlying native layer ISPs in order to offer a value-added network service to actual end-systems,

and ii) end-system overlays (e.g., Skype[11]). In the presence of filtering, the user experience will suffer in both these overlays. However, because a service overlay is managed by a single operator, it is feasible for the overlay network to regain the full advantage of overlay routing by adopting one of two approaches we propose. In the first approach, overlay nodes are added so that good overlay paths do not represent inter-domain policy violations. This approach attempts to insure that overlay paths conform to native policy. In the second approach, the overlay acquires transit permits or exit permits from certain ASes that allow certain policy violations to occur but only for permitted overlay traffic. This approach attempts to "legitimize" native policy violations through commercial agreements between the overlay service provider and the native network.

We further develop a single *cost-sharing* framework that allows the incorporation of both these approaches into a single strategy. We formulate an optimization problem that aims to determine how the overlay network should allocate a given budget between paying for additional overlay nodes and paying for transit permits to ASes. We develop a heuristic solution to this problem and illustrate its application on our overlay case study. Further, we evaluate its performance under varying network characteristics.

The remainder of this chapter is organized as follows. We define and describe the different types of policy violations possible in Section 4.2 and classify them in Section 4.3. We characterize the extent of native policy violations using our case study overlay network and present associated results in Section 4.4. Section 4.5 investigates the effect of native layer enforcement of routing policy on the performance of the overlay. We present our cost-sharing approach for mitigating the effect of packet filtering in Section 4.6. Previous research related to our work are briefly described in Section 4.7. We summarize this chapter in Section 4.8.

4.2 Description of Policy Violations

The current Internet is made up of thousands of autonomous systems that coexist, cooperate, and compete for usage of its various resources. Each AS establishes some form of native layer policy to express its willingness to allow or deny traffic from its neighboring

ASes. The Border Gateway Protocol (BGP) is the policy-based routing protocol that runs between autonomous systems, implements the various policy constraints and helps each AS select the routes to a destination.

The native network policies are primarily motivated by economic costs and performance gains[115]. These policies predominantly reflect the commercial agreements between ASes and are encoded into the router configuration to be enforced at ingress and egress points of the administrative domain. The combination of these individual policies determine the *AS-path* used by the flows in the Internet; the AS-path is defined as the ordered list of ASes a packet needs to traverse to reach the intended destination. The interconnection of over 20,000 different ASes in the Internet leads to a complex structure with path inflation[151] and unpredictable routing behavior at times[102].

4.2.1 Definition of Policy Violations

We define policy violation as the act of using a route at the overlay layer that actually may be objectionable to the ASes involved if used at the native layer. The only reason overlay layer is able to use such a route is because of the misdirection created by overlay relaying, which hides the actual destination from the native layer.

4.2.2 Types of Policy Violations

There are different types of native layer policy violations possible, based on what policies the border router enforces. Overlay routing can potentially violate any of these policies at will. We study the violation of two forms of inter-domain policy:

- *Transit policy*: The transit policy (also known as the *valley-free* policy) of AS-paths states that no AS will act as a transit for traffic originating from its provider or its peer, unless the traffic is destined to its customer[59]. We are interested in the violation of the valley-free property because this is the case in which the violation is experienced by an AS not involved in any way with the end-to-end communication. Hence, we consider this to be the most reproachable.

- *Exit Policy*: The exit policy is modeled as *the preferred combination of the next hop*

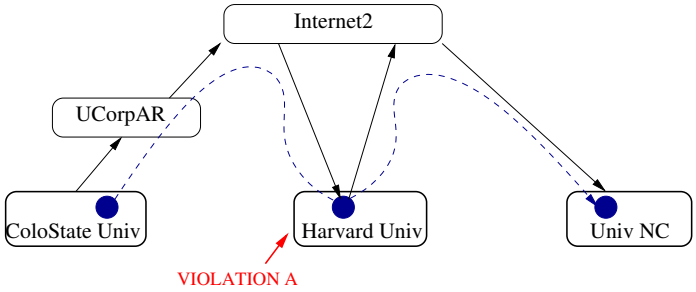


Figure 9: Example of a plausible violation in reality. The solid circles represent the overlay nodes and the dashed line represents the end-to-end overlay path.

AS and the egress inter-domain link, for a particular destination IP prefix. We define exit policy violation as a deviation from this preferred exit caused by overlay relaying. We posit that such deviation causes undesired load and expenses for the native layer, and is objectionable to the native ASes. However, the level of objection to each exit violation may vary with each AS and each overlay path.

Fig. 9 gives an example of both forms of native layer policy violation we noticed during our overlay route measurement process. In this example, the overlay path shown in the figure was determined to be optimal and in particular was superior to the native network path from Colorado State University to the University of North Carolina. We observe that Harvard University is having to use its access bandwidth to/from the Internet to act as a transit between Colorado State University and the University of North Carolina. Thus, the situation might be undesirable for the Harvard University native network, while being beneficial to the overlay nodes at Colorado State University (by providing it an alternate path with 6.3ms lower latency, a 10.48% gain over the native route). We also observe that Internet2 is forced to use different inter-domain links for sending traffic destined to University of North Carolina. This may be objectionable if it was not previously compensated for that usage.

Note that if the overlay node at Harvard University were also a *consumer* of the data being forwarded (in addition to being a relay), we do not consider this a violation since this becomes true application-layer forwarding. In the above example, if the overlay node at Harvard University were a consumer of the data, then it will be part of the communication

even in the absence of relaying. More examples of this include end-system multicast, email forwarding and P2P file-sharing, where the intermediate node also uses the content. Hence, a transit violation is when the actual end-to-end AS-path used by the overlay is a violation of the native routing policy and the relay nodes on the overlay path are non-consumers of the data being forwarded[1].

4.2.3 Economic Model

In this chapter, we assume that each overlay node is placed in an AS i by paying a new node fee of N_i ($N_i \geq 0$). However, each added overlay node is bound by certain end-user agreements with the hosting AS, which clearly specifies the allowed access. We adopt such a model because the hosting AS in turn incurs expenses (monetary and load) to provide connectivity to this overlay node.

An example of this agreement is one where Internet2 specifies that its customers do not use their network for commercial traffic. As mentioned earlier, overlay routing is quite capable of violating this agreement (or policy). Another example of a violation is when an overlay node within an AS starts providing Internet connectivity to other home users by running a PPP connection over the phone line and surreptitiously forwarding the associated traffic to the Internet. Such examples reveal that the exact policy violation is when the overlay node uses the native network for more than it agreed for.

Consider the example in Fig. 10 where overlay nodes are shown as customers of the native network. Although all overlay nodes form a single administrative domain at the overlay layer, they are part of three different administrative domains at the native layer. This causes objections to anarchical behavior from the customer nodes and might cause the hosting AS to revisit the end-user agreement.

In Fig. 10, assume node B connects to Client_2's network by paying a fee of $\$C_B$. This can be for 1) unlimited usage, 2) certain amount of bytes expected from B, or 3) certain access bandwidth consumed by B. Note that Client_2 in turn pays Provider_1 and Provider_2 for its Internet connectivity. Hence, the contract with node B will be supplemented by a

[1]Our work is restricted to scenarios that are confirmed to be an instance of overlay routing, although it is complicated to verify if the traffic is relayed.

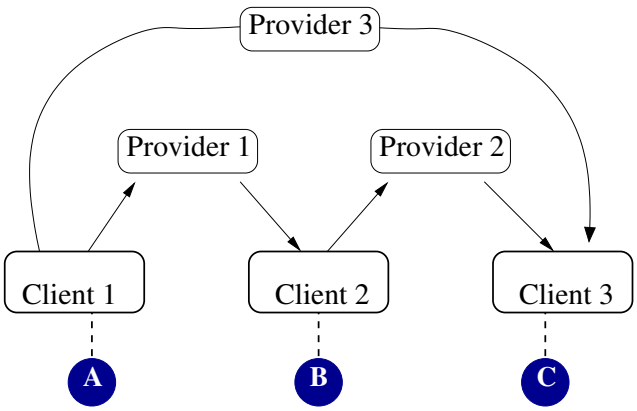


Figure 10: Illustration of connectivity between overlay network and native network. The solid circles represent the overlay nodes and the dashed line represents the end-user agreements.

end-user agreement that states acceptable behavior because Client_2 wishes to keep its cost incurred low. In such a scenario, the overlay node B acting as a transit between node A in Client_1 and node C in Client_3 may cause unnecessary expenses for Client_2, for certain end-user contracts. This is true in the case of academic ASes where the overlay node has unlimited usage rights. This explains why violating transit policy at the overlay layer can be objectionable. Furthermore, Client_1 may possibly have to use a more expensive ISP Provider_1 for traffic that is actually destined to node C. This may be undesirable from the perspective of Client_1. This explains why violating exit policy can be objectionable.

The level of objection may be different with each violation. It is possible that for certain end-user agreements there may be no objections from the native layer. In the rest of the chapter, however, we assume the worst case scenario, where all violations are serious and are of equal importance to the ASes concerned.

4.3 Classification of Policy Violations

In this section, we present our classification of the various forms of transit and exit violations, and highlight some important observations about their nature. Further, we describe the relation that exists between two classes of violations.

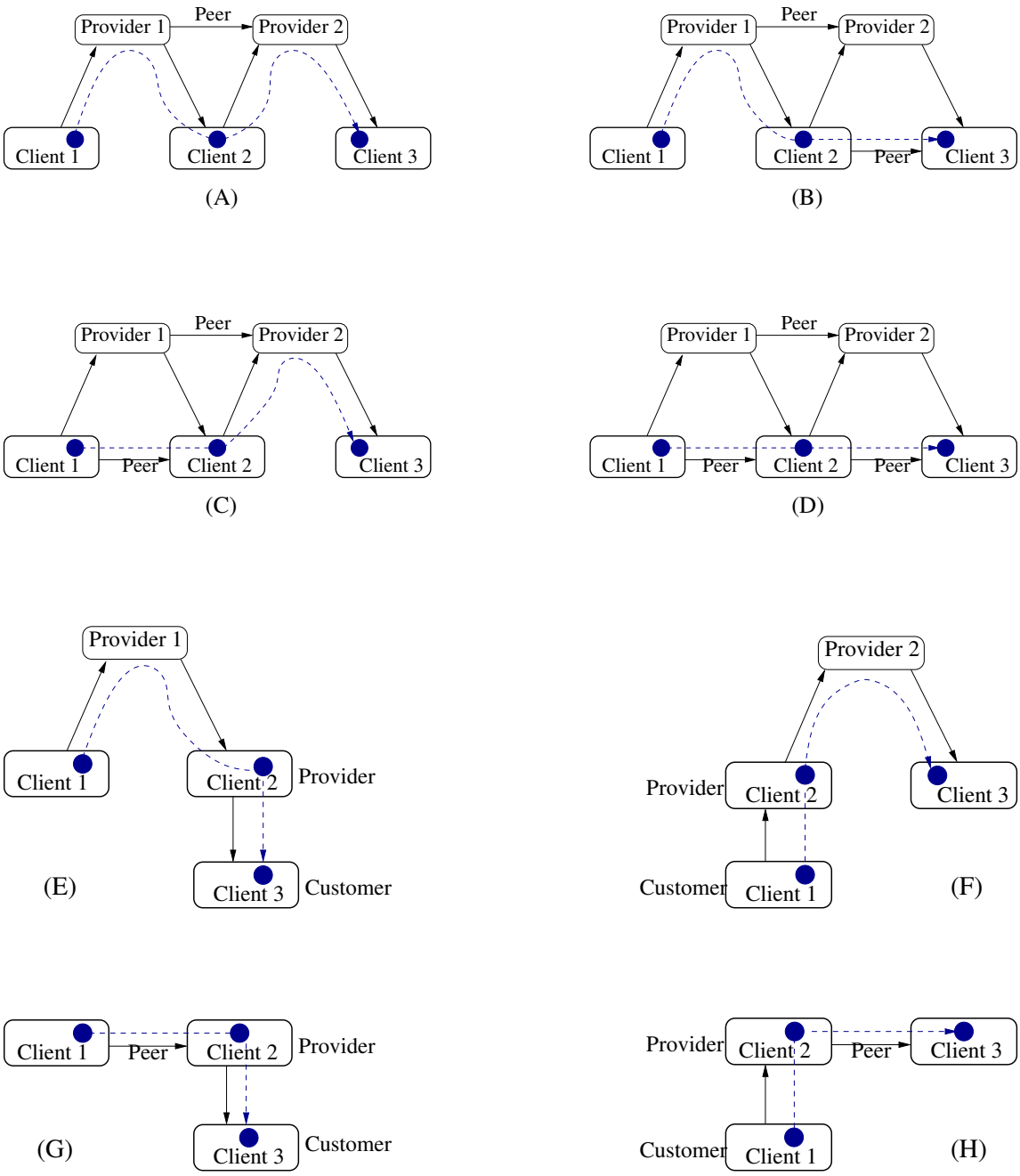


Figure 11: AS relationships within each overlay path. The solid circles represent the overlay nodes and the dashed line represents the end-to-end overlay path.

4.3.1 Classification of Transit Violations

Previous studies on BGP misconfiguration highlighted four different types of route export violations prevalent in the current Internet[102], each of which violates the valley-free property of AS-paths. In the same spirit, we investigate the type of violations caused by overlay routing, which control the route for a certain end-to-end connection by using intermediate relays.

Fig. 11 illustrates eight different forms of relaying possible. We argue that cases A, B, C and D represent a violation of native layer policies as the overlay traffic uses the intermediate overlay node to transit through client 2. Thus, client 2 acted as a transit between its provider and peer, which is a violation of the commercial relationships between the ASes. However, cases E, F, G and H do not represent violations because one of the overlay nodes was located in a provider (non-stub) network. In general terms, no violation exists if the native routing policy condones or allows client 2 to be a transit between client 1 and client 3.

4.3.2 Classification of Exit Violations

Fig. 12 illustrates the four basic forms of exit policy violations possible. We describe each as follows:

E1. *Next hop AS violated*: This is caused when the overlay traffic is relayed through an intermediate node located in an AS not along the direct native route between the end nodes. Fig. 12(E1) illustrates a simple scenario where the source overlay node causes Client_1 to pick Provider_2 to indirectly reach a destination in Client_2.

E2. *Ingress or Egress router preference violated*: An exit point violation can happen when an overlay path uses a relay node in a downstream AS that is closer to a different ingress router not used by the direct native route. Clearly, this could be a violation of the Localpref attribute, hot-potato routing or cold-potato routing. In Fig. 12(E2), we can see that the local egress preference (router $R1$) and the neighbor's ingress preference (router $R3$) are violated, without the knowledge of either AS. This has the

problem that the load from the overlay traffic is borne by the link $R2 - R4$ instead of the designated inter-domain link $R1 - R3$.

E3. *Exit point violated because of a next hop AS violation*: This form is similar to the previous scenario, except that the router preference is affected by the alteration of the next hop AS at a downstream provider. In Fig. 12(E3), we see that a change in next hop AS at Provider_1 causes a change in the ingress point from $R3$ to $R4$. Such a change in ingress router is achieved when Provider_1 offers a different MED value for its perceived destination prefix.

E4. *Next hop AS violated because of an exit point violation*: When a downstream AS spans a wide geographical region, it is possible that a change in the ingress point into its domain might cause it to alter its preference of the next hop AS. In Fig. 12(E4), we see that a change in the ingress point from $R3$ to $R4$ causes Provider_1 to transit traffic through Provider_2, rather than sending directly to Client_2.

The above four scenarios capture the different types of exit policy violations. Each violation has serious economic or load repercussions and is undesirable from the perspective of the native service provider.

The exit violations in each path originate at a single AS where the inter-domain link changes. However, based on the particular type, it may be experienced by other downstream ASes as well. Unlike valley-free violations, the exit violations are not restricted to only an intermediate host AS[2].

4.3.3 Preliminary Observations

There are multiple reasons for using a multi-hop overlay path, rather than the direct overlay link (essentially the native network path between two nodes) e.g., achieve better performance or resilience, circumvent limitations imposed by firewalls and NAT boxes, or load balancing, to name just a few. Note that the basic native route between two overlay nodes (referred to as the *overlay link*) never constitutes a policy violation; under the assumption

[2]We refer to an AS in which an overlay node is located as the *host AS*.

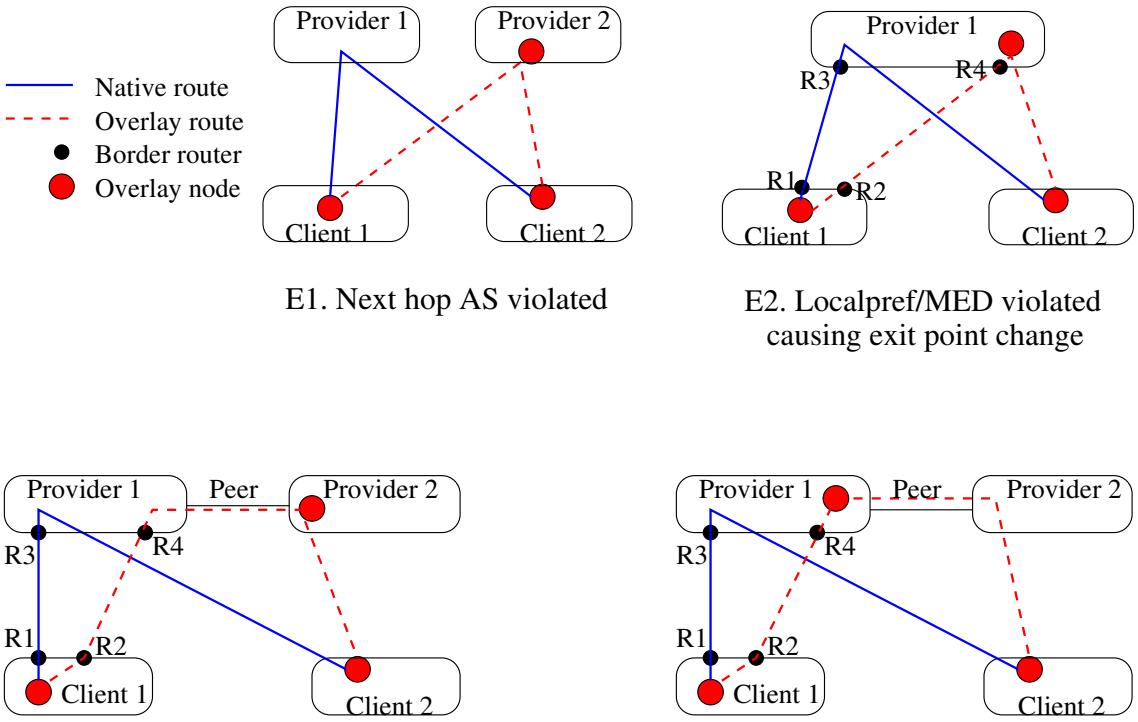


Figure 12: Possible exit policy violations. Each of these violations can occur at any point of the multi-hop path, either at a host AS or a non-host AS.

that the native routing conforms to policy. Thus, we make the following observation and investigate only multi-hop overlay paths in the rest of the chapter:

> **Observation 1.** *A single-hop overlay path between two overlay nodes does not represent a native transit policy violation.*

An interesting artifact of shortest path routing performed at the overlay layer is that *it is sufficient to analyze each 2-hop overlay path.* Consider an overlay network with nodes A, B, C and D. If the shortest path between nodes A and D is $ABCD$, then the shortest path between nodes A and C is ABC. Hence, if the 2-hop overlay path ABC or BCD is violating, then the 3-hop overlay path $ABCD$ will also be violating. Furthermore, the number of violations in a multi-hop path is a summation of the violations observed in its constituent 2-hop overlay paths. This confirms that the 2-hop overlay path scenarios considered in Fig. 11 and Fig. 12 represent all types of policy violations possible.

Based on the discussion about which scenarios represents a transit violation and which do not, we make the following observation:

> **Observation 2.** *When the overlay nodes are located in stub domains (domains with no clients), every multi-hop overlay path in which the relay nodes are not data consumers represents a transit policy violation.*

From the classification of exit violations, we make the following observation:

> **Observation 3.** *An exit violation can originate at any one of the three following locations:*
>
> - *Source AS (which is also a host AS)*
> - *Intermediate host AS*
> - *Intermediate non-host AS*

4.3.4 Relation between the Two Classes of Violations

We argue that any overlay path having a valley-free violation necessarily has at least one exit violation at an upstream AS. This can be reasoned by the fact that a valley-free violation occurs at an intermediate host AS that lies outside of the direct native route and such a deviation must originate at an upstream AS (See Fig. 8, for example). Thus, exit violating paths represent the superset of all violating paths. Furthermore, in a particular multi-hop overlay path, the same AS will never experience both exit violations and valley-free violations. This is because valley-free violations happen only at ASes that are not along the legitimate route between the end points, in contrast to exit violations.

We address the following two important questions in the next two sections:

- What is the extent of native layer policy violation in a typical overlay routing situation? - Section 4.4
- If the native routing policies are enforced and we disallow certain routes, how much is the overlay routing efficiency affected? - Section 4.5

4.4 Characterizing Overlay Violations

In this section, we provide insights into the extent of native policy violations in overlay networks. We do this using an experimental case study overlay network deployed over Planetlab[121]. We first describe the overlay case study and investigate the characteristics of its overlay paths. Next, we evaluate the performance gains that overlay routing provides. Lastly, we examine the extent of policy violations that show up in the case study. The overall measurement methodology is summarized in Fig. 13.

4.4.1 Overlay Network Case Study

We choose 58 Planetlab nodes that are geographically distributed (based on latitude/longitude) over North America, with only one node per AS. We refer to the AS in which an overlay node is located as the *host AS*.

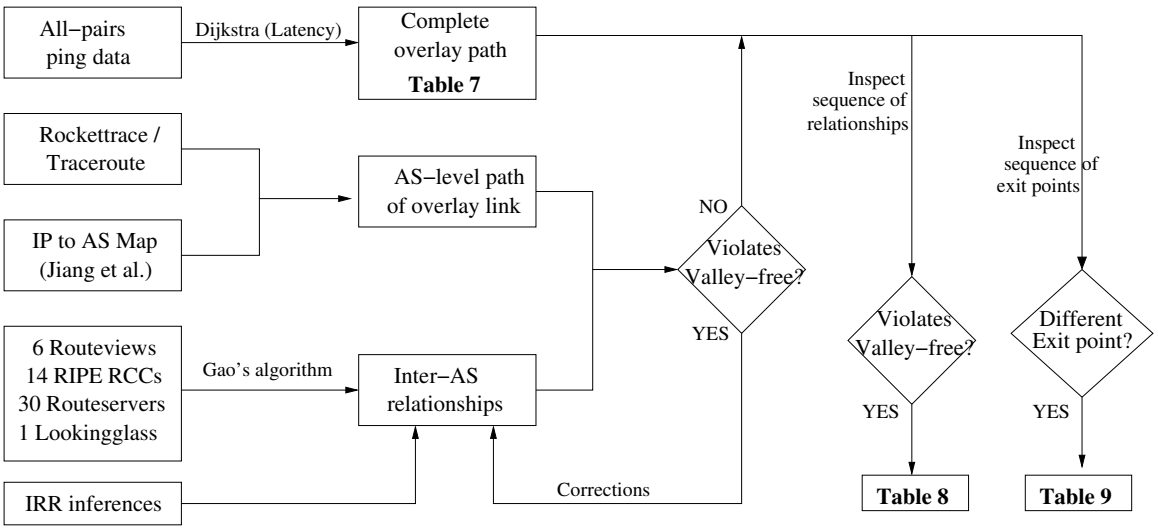


Figure 13: Measurement methodology

We assume complete mesh connectivity of overlay links between the overlay nodes, for all the results presented in this thesis. Following standard terminology, an *overlay link* represents the direct native route between two overlay nodes, which in turn comprises one or more native links, and an *overlay path* is made up of one or more overlay (virtual) links. This overlay path represents the end-to-end route taken by the overlay application traffic. There are a total of 3306 overlay paths possible in our topology.

The best overlay path between two overlay nodes may not always be the direct path. In many cases, it is beneficial to route the overlay connection through other overlay nodes, than adopt the direct route[6, 51]. Such a decision is typically made by running a routing algorithm at the overlay layer using application-specific routing performance objectives.

For the case study overlay network, we ran a shortest path routing algorithm using hop counts and latency. Table I(a) presents the different multi-hop overlay paths observed. Table I(b) further classifies such multi-hop paths when the routing metric is latency. It is interesting to note that almost 56.5% of the overlay paths use a multi-hop route. This provides us ample data to analyze. Moreover, end-to-end latency is a metric that many applications would like to optimize. Hence, through the rest of the case study analysis, we study the violations observed for the particular routing metric of latency.

Table 7: Multi-hop paths in our Planetlab measurements

Metric	**Measurement Scheme**	**# multi-hop paths (of 3306 total paths)**	**%**
Hop count	Scriptroute[145]	394	11.91
Latency	Ping RTT	1868	56.50

(a) Summary of number of multi-hop paths

Direct	**Hop count of multi-hop paths**						
	2	**3**	**4**	**5**	**6**	**7**	**8**
1438	1053	375	315	85	20	17	3
43.5%	31.9%	11.3%	9.5%	2.6%	0.6%	0.5%	0.1%

(b) Latency-based multi-hop paths

4.4.2 Overlay Routing Performance

Here we address the performance improvement obtained in our case study through the use of multi-hop overlay routes. We quantify the efficiency of overlay routing by using the *gain* metric. The gain achieved for a path is defined as:

$$\text{Gain for path AB} = \frac{(\text{Overlay link latency})_{\text{AB}} \text{ - } (\text{Overlay path latency})_{\text{AB}}}{(\text{Overlay link latency})_{\text{AB}}}$$

where the $(\text{Overlay link latency})_{\text{AB}}$ is the latency of the direct native route between nodes A and B, and $(\text{Overlay path latency})_{\text{AB}}$ is the latency of the shortest path through the overlay network between nodes A and B. Note that $(\text{Overlay link latency})_{\text{AB}} \geq (\text{Overlay path latency})_{\text{AB}}$ always.

The gain metric represents the reduction in end-to-end latency achieved relative to the native route. The value ranges between 0, when the direct overlay link is the optimal one, and 1, when the multi-hop overlay path latency (which is the optimal one) is very small relative to the direct path.

Fig. 14 plots the values of gain observed for each multi-hop overlay path in our data set, sorted based on the hop count of the overlay path and ordered in the increasing order of gain. In the same graph, we plot the corresponding hop count of that overlay path. We observe that the individual curve for each hop count is similar and comparable. This indicates that in our case study there is not much dependence between the hop count and

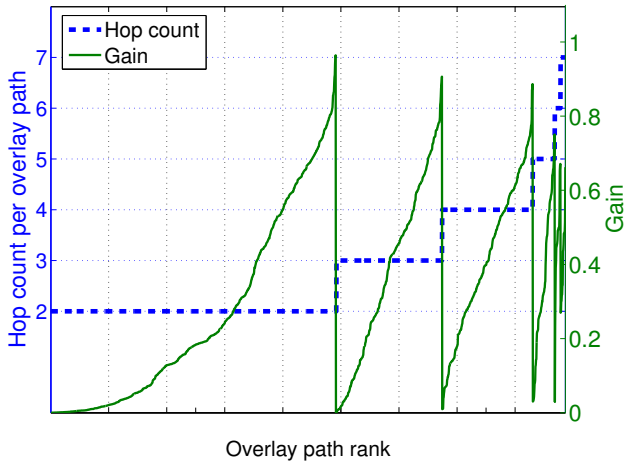


Figure 14: Relation between gain observed for each overlay path and the overlay path hop count.

the gain. In other words, a higher hop count does not indicate a lower gain as one would guess.

The *betweenness* of a node is the number of overlay shortest paths that pass through it[56]. Fig. 15 plots the values of (non-zero) betweenness that were observed for each overlay node in our data set, sorted based on the decreasing value of betweenness. We clearly observe a non-uniformity in the relay popularity. From the shape of the betweenness curve, we conclude that there are a few overlay nodes that are the main reason for multi-hop overlay paths being preferred over the direct route. In the figure, we also mark the nodes located in stub domains and those located in non-stub domain. We can note that our dataset has a number of non-stub domain nodes with high betweenness. This is the main reason for the high percentage of non-violating overlay paths (as seen in the following subsection). However, the number of overlay nodes located in non-stub domains in our dataset is not restricted to those indicated in the figure, as the figure plots only nodes with non-zero betweenness that are currently being used in the overlay paths. In Fig. 15, we also plot the value of out-degree for its host AS and its siblings. We do not observe a particular correlation between the betweenness and the AS out-degree, which indicates that overlay routing is strictly driven by the latency-based edge costs and not the number of AS neighbors.

4.4.3 Estimation Methodology

Estimating the extent of policy violations in our case study requires the end-to-end AS-level path of each overlay path and the individual relationship between each consecutive pair of ASes in the end-to-end AS-path.

4.4.3.1 Inferring AS Path

The Scriptroute[145] data from the Planetlab measurements provides the IP address of each native hop in the overlay link[3]. We use the publicly available IP-prefix-to-AS mapping generated by the dynamic algorithm in [103]. This work primarily extracts the origin AS of each IP prefix from the BGP routing tables (from sources like the Routeviews servers[131], RIPE servers[128]) and refines the entries further using a dynamic algorithm. By performing a longest prefix match, we obtain the IP-to-AS mapping for each of the native hops. We also cross-verified our IP-to-AS mapping with those generated by *undns* in the Scriptroute tool.

After resolving the AS number of each IP hop in the overlay link, we have the end-to-end AS-path for each overlay path. Any usage of the term AS-path, henceforth, will represent the sequences of ASes in an overlay path, derived by concatenating the AS-path of individual overlay links, unless specified otherwise.

4.4.3.2 Obtaining AS Relationships

In order to obtain AS relationships, we adopt Gao's algorithm[59], supplemented by the partial AS relationship information[166], and our own heuristics to eliminate most of the algorithm's inaccuracies. Gao's algorithm makes inferences based on the AS-paths extracted from the BGP tables and identifies each relationship as being either a customer-provider relationship, a peering relationship, or a sibling relationship. The output from Gao's algorithm is more accurate when we input a more complete view of the AS-paths currently used in the Internet. Hence, as suggested in [172], we obtained the BGP tables from 6 RouteViews servers[131], 14 RIPE RCCs[128], 30 public routeservers and 1 lookingglass

[3]In certain anomalous cases, where the rockettrace did not work at certain nodes, we performed the traceroute operation.

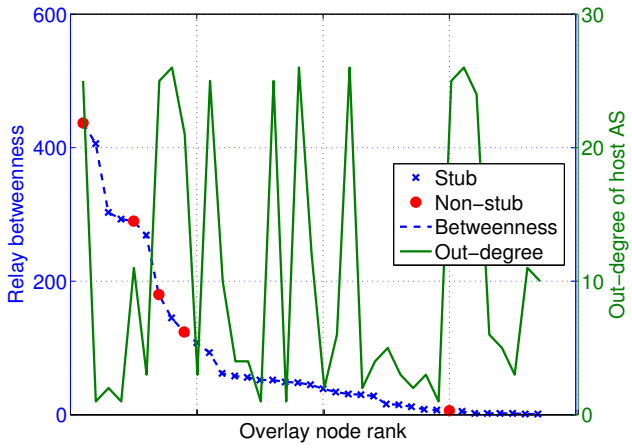


Figure 15: The betweenness of each overlay node, plotted along with the out-degree of its host AS.

server[99].

As the algorithm is forced to use heuristics to classify some of the ASes as a provider, the relationships inferred are not guaranteed to be error-free. Firstly, it has been established that Gao's algorithm does not have a good level of accuracy with inferring peering and sibling relationships[166]. Secondly, the BGP table does not necessarily have information about all the possible inter-AS connections, as some ASes (representing stub networks that are most often simple customers) do not export routes to its peers. To solve these issues, we apply three corrections to the output of Gao's algorithm:

- Partial AS relationship information extracted from the RADB and the RIPE databases of the Internet Routing Registries (IRR)[75]. We followed a procedure similar to that in [166] to obtain these partial relationships.

- Implications from observation 1 that *the AS-path of all overlay links must be valley-free*. For instance, Gao's algorithm inferred that Global Crossing is a customer of Level3 Communications. However, this inference violated the valley-free property of the AS-path of some overlay links. Hence, we resolved the relationship according to what might lead to the valley-free property.

- Hypothesis that unknown relationships between a pair of stub ASes are often unreported peering relationships.

Table 8: Analysis of overlay paths for transit policy violations

Type	Description	Number	%
A	Provider-AS-Provider	1925 relays	63.09
B	Provider-AS-Peer	74 relays	2.43
C	Peer-AS-Provider	61 relays	2.00
D	Peer-AS-Peer	73 relays	2.39
None	No violation	918 relays	30.09

(a) Summary of violating relay operations

Type	2	3	4	5	6	7	8
A	41.7	64.7	75.1	76.6	77.6	77.3	80.0
B	5.3	2.7	0.8	0.0	0.0	0.0	0.0
C	3.0	3.4	1.1	0.0	0.0	0.0	0.0
D	5.6	1.0	0.1	3.9	1.0	0.0	0.0
None	44.4	28.2	22.9	19.5	21.4	22.7	20.0

(b) Split up of transit violations for paths of a certain hop count

4.4.4 Transit Policy Violations Observed

We used the AS information obtained above to characterize the transit policy violations in our case study. The statistics are summarized in Table II(a). Note that each overlay path might commit multiple native policy violations, thereby giving a total of 2629 violations for 1868 multi-hop overlay paths. From the table, we observe that a predominant portion of the violations are of type A (as described in Section 4.3), followed by those of type B. It is also worth noting that 30.09% of the 3115 intermediate relaying operations performed by the multi-hop overlay paths do not constitute a violation.

We also observed that about 30.19% of the 1868 multi-hop overlay paths do not commit any native policy violation. This is because all intermediate overlay nodes of those paths are located at a non-stub AS. In our dataset, all multi-hop overlay paths with more than 2 relays (3 hops) represent a violation.

Table II(b) shows the individual percentage of violations for overlay paths of different length (in terms of the overlay hop count). It is interesting to note that overlay paths with high hop count display more violations of type A, and rarely any of the other three

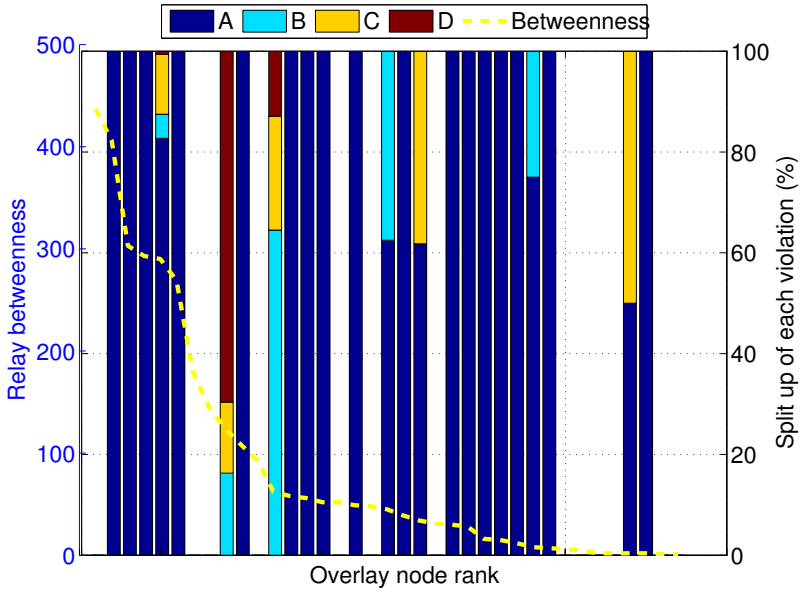


Figure 16: Partitioning of the 4 different policy violations present at each relay

types, implying that peering relationships are rarely present in long overlay paths in our case study. This could be because the peering relationships do not always offer a path with better latency, but rather offer paths with lower economic cost.

Fig. 16 presents the partitioning of the different transit violations observed in the domains that host the intermediate overlay nodes, along with the betweenness of the corresponding overlay node. We observe that most of the overlay nodes with high betweenness have a non-zero number of policy violations and most of these violations are of type A. We can observe from Fig. 15 and Fig. 16 that overlay nodes located at non-stub ASes do not commit native policy violations.

4.4.5 Exit Policy Violations Observed

Using the same measurement data, we estimated the frequency of exit violations noticed by comparing the shortest path and the direct native path in the original Planetlab data. Table 9 presents the summary of exit violations observed. We note that nearly 87.7% of the multi-hop overlay paths represent an exit violation. Interestingly, not all multi-hop overlay paths represent an exit violation as one might expect. This is because the AS path and the exit routers used are the same for almost 12.2% of the overlay paths, though the exact

Table 9: Exit violations noticed in the Planetlab dataset, along with the number of paths having valley-free violations.

Type	Originating Location	Exit violations	%	Valley-free violations
E1	Source AS	502	26.9	444
	Intermediate host AS	104	5.56	76
	Intermediate non-host AS	372	19.9	331
E2	Source AS	43	2.30	1
	Intermediate host AS	113	6.05	0
	Intermediate non-host AS	135	7.22	11
E3	Intermediate host AS	26	5.83	19
	Intermediate non-host AS	342	18.3	326
E4	Source host AS	1	0.05	0
Total violating paths		1638	87.7	1208

route traversed by the multi-hop overlay path is indeed different from that of the direct native route. This peculiarity is mainly observed when the intermediate node is located in the Internet2 AS. We also affirm from the data that all 1208 multi-hop overlay paths having valley-free violations also have exit violations.

Most of the exit policy violations we observed in our testbed network was of the type where the next hop AS is changed by overlay routing, viz. $E1$ and $E3$. Note that for violations of type $E3$, we consider the change in the next hop AS as the most reproachable and mark the AS experiencing it as the origin point. Hence, we do not see a source AS originating the violation $E3$. Furthermore, we see more exit violations originating at source ASes (33.2% of total) and at intermediate non-host ASes (51.8% of total) in our dataset.

As mentioned earlier, a violation in a 2-hop overlay path BC is potentially seen in each overlay path AD that overlaps path BC i.e., the same exit violation might be part of two different paths because the end points are different, although the intermediate segments are the same. Furthermore, a violated source AS in the path BC will count as a violated intermediate host AS for all multi-hop paths AD that overlap this particular path. Note that the results presented above correspond to the total number of violating paths and not number of unique exit violating hops.

Fig. 17 presents the number of exit policy violations experienced by each AS traversed

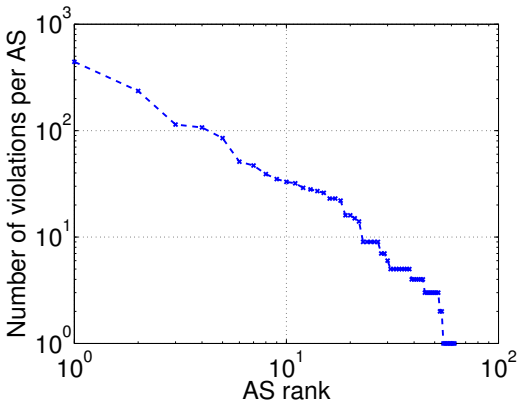


Figure 17: Log-Log plot of the number of exit violations experienced by each of 62 ASes. This shows clear non-uniformity in the violations experienced.

by the multi-hop overlay paths. We observe that the number of exit violations is non-uniformly distributed, such that a small fraction of the ASes is violated by a large number of overlay paths.

Lastly, we analyze the relation between the transit violation and exit violation. When we inspect all exit violations that happen in the overlay link AB because of a transit violation at node B, we notice a stronger correlation between the AS experiencing the most exit violations and the node B, rather than with node A. However, a particular transit violating intermediate relay does not have a unique exit violated AS preceding it i.e. there is no one-on-one correspondence between the transit violated AS and the exit violated AS. The number of exit violations per AS, associated with a particular intermediate relay, is often distributed in the same non-uniform manner as in Fig. 17, i.e. Some ASes experience significantly more exit violations compared to others. This observation helps us improve the mitigation strategy in Section 4.6.

Although this thesis only considers the metric of latency at the overlay layer, we expect similar violating behavior with other metrics (e.g., bandwidth) as well, as long as the overlay routing offers substantial improvement over native routing. This can be reasoned by the fact that policy violations are primarily caused by multi-hop overlay paths, which tend to deviate from the direct native route. Thus, the higher the number of multi-hop overlay paths (in other words, higher the percentage of relaying), the higher the extent of policy

violations.

It is possible that a certain deviation from the standard end-to-end path may not be objectionable to the AS involved. In our dataset, we observed that 61.05% of the deviating overlay paths (43.83% being of type $E1$) have an AS path that is substantially longer or shorter than that of the direct native route. In such cases, we can assert that the exit violations observed are indeed objectionable. We can say so because an AS that has a choice between two inter-domain routes will first pick based on policy and then based on length. Hence, choosing a path that is substantially different in exit points and AS length is clearly a violation of both decision criteria and definitely objectionable. However, we are unable to establish with confidence the seriousness of the other violations.

4.5 Effect of Filtering on Overlay Performance

In Section 4.1, we briefly described the motivation behind various administrative domains (AS) aiming to filter overlay layer traffic. Such filtering is propelled by the negative impact of overlay routing on the native layer, like: defeat of traffic engineering[98, 83], route instability[98, 83, 138], and eventually upsetting the economics of AS interconnection.

This filtering may defeat the purpose of overlay networks and eliminate the flexibility in overlay routing. Nevertheless, it is essential that the native layer have some basic control over such cases of policy violation and then exercise its discretion in determining what can or cannot be allowed. This need is illustrated by the number of commercial solutions that provide such overlay traffic filtering functionality[161, 117, 144].

We start with the premise that native network filtering is possible (Refer to Section 4.7 for a survey of existing strategies) and consider two types of filtering – 1) *Blind filtering*, in which an AS blocks all overlay relaying through it, and 2) *Policy-aware filtering*, in which an AS blocks transit overlay traffic only if it violates native routing policies. Note that blind filtering may be easier to implement since it does not require knowledge of native routing policy. In either case, ASes filter only relayed (multi-hop) traffic and do not filter overlay traffic that use the direct native route.

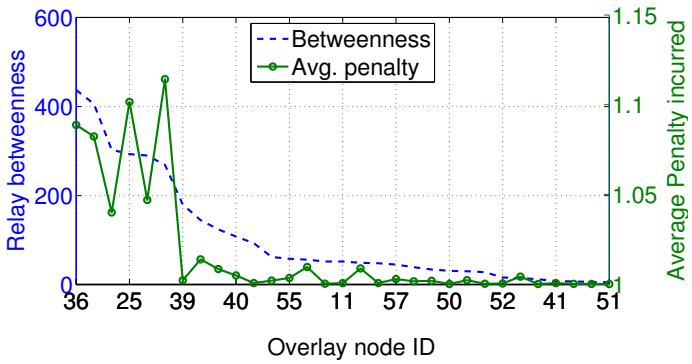


Figure 18: Performance when overlay relaying is blindly disallowed at a particular host AS, one AS at a time

When all ASes perform blind filtering, we observe that no overlay path can take a multi-hop overlay route between two end-points. This makes it essential that all overlay nodes are mesh-connected to ensure that all nodes can reach each other. However, when all ASes perform policy-aware filtering and disallow all types of transit policy violations, we notice from results in Section 4.4.4 that some (30.19% in our dataset) of the multi-hop overlay paths, which commit absolutely no transit policy violation, are not blocked. Hence, we still maintain the benefits derived from overlay routing. This situation is more desirable for the overlay network and its users. However, if exit violations are also disallowed, then the overlay performance suffers drastically as only few (12.2% in our dataset) of the multi-hop overlay paths are not blocked.

We now evaluate the impact of these two forms of filtering on overlay routing using a *penalty* metric. The penalty incurred for each path is defined as:

$$\text{Penalty for path AB} = \frac{\text{Post-filtering latency of overlay path AB}}{\text{Best possible latency of overlay path AB, assuming no filtering}}$$

The penalty value is a good indicator of the negative effect when an intermediate relay node in the shortest overlay path disallows relaying and the overlay traffic is forced to take a longer path.

We next use our Planetlab overlay, characterized earlier, as a case study to evaluate the effect of filtering on overlay routing performance.

4.5.1 Blind Filtering

In Fig. 18, we plot the average penalty (averaged over all multi-hop overlay paths) that would be incurred if the host AS of a particular overlay node blindly filters out overlay traffic, alongside the value of betweenness associated with that overlay node. To compute the value of the penalty for a particular overlay node, we rerun the shortest path algorithm after disallowing relaying at that node and compute new end-to-end latency values. When an overlay node with high betweenness is disallowed, we seem to incur a high penalty. This indicates that few overlay nodes with high betweenness provide a substantial incentive to each overlay path passing through it and are quite irreplaceable. Hence, depending on whether an excessively used relay is being disabled the overall penalty incurred varies.

Fig. 19 presents plots for the penalty incurred and the number of violations committed when the number of ASes performing this filtering operation is varied. When the number of host ASes performing the filtering operation is n, we have $\binom{58}{n}$ possible scenarios. In cases where this number is over 1000, we chose 1000 scenarios at random. Each penalty measurement in Fig. 19 is an average of the penalty incurred by all overlay paths, and average over all scenarios considered. We also plot the 95% confidence interval for each value.

We observe, from Fig. 19 that, as expected, when more ASes hosting overlay nodes perform blind filtering of overlay traffic, the penalty incurred increases, while the number of violations decreases. We also observe a drastic drop in the number of violations as more than 50 host ASes begin filtering because the last few ASes with high betweenness are finally being targeted.

When all 58 host ASes begin filtering, the overlay paths are forced to use the single-hop overlay link without any relaying, which is equivalent to native routing. This leads to the following three consequences:

- The number of violations observed is 0.
- The penalty value is at a maximum of 1.838.
- There is no advantage to using overlays, as the overlay path is the same as the direct

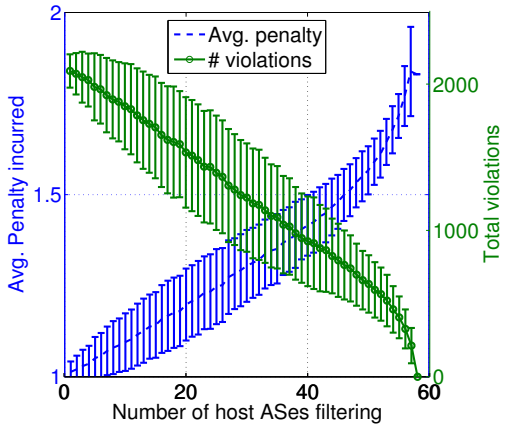


Figure 19: Blind filtering

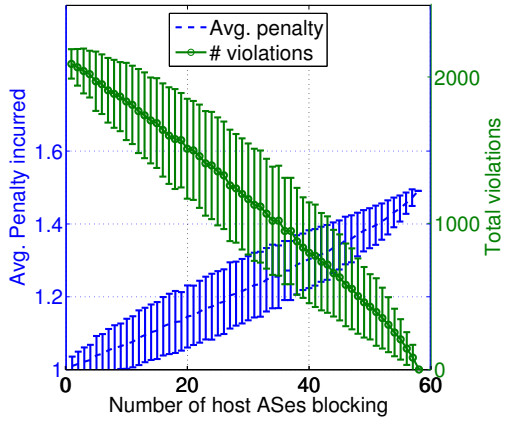


Figure 20: Policy-aware filtering

native route.

4.5.2 Policy-Aware Filtering

We enforce transit policies on the overlay path by filtering at ASes that host the intermediate overlay nodes of a multi-hop overlay path. We present only the impact of enforcing transit policies because we consider them most reproachable. Furthermore, the impact of enforcing exit policies is similar to that of blind filtering, albeit a bit less detrimental.

Fig. 20 presents plots for the penalty incurred and the number of violations prevalent, along with the 95% confidence interval for each value, when native transit policy violations are disallowed by a certain number of host ASes. We use the same evaluation methodology as that described in the previous subsection. Similar to blind filtering, the penalty incurred increase and the number of violations decrease, with an increase in the number of host ASes performing the policy-based filtering.

When all 58 host ASes begin filtering, the violating overlay paths are forced to use the direct route without any relaying. Hence, the number of violations is 0. However, this does not imply that the gain is 0, as some multi-hop overlay paths are still allowed. In our dataset, we observe an average gain of 13.49% in this scenario (in contrast to 31.81% in the case where there was no filtering). The penalty value is at a maximum of 1.49.

When we compare the results in Fig. 19 and Fig. 20, we notice that policy-based filtering incurs substantially lower penalty compared to blind-filtering and a non-zero gain,

making it still worthwhile to deploy overlays.

4.6 Mitigating the Impact of Filtering

In the previous section, we investigated the effects of policy enforcement at the native layer. When the overlay traffic relaying is disallowed at the native layer, we notice a substantial deterioration in performance of the overlay networks. This incapacitation causes the overlay traffic to experience the same or worse treatment as traffic generated by other applications. It is also conceivable that the overlay network avoids using overlay paths with native policy violations in an effort to appease the underlying native network. This tends to have the same drop in overlay routing gain as filtering does.

In the context of service overlays that are managed by a common overlay service provider (OSP), we propose two possible options that can allow the overlay regain some of its performance advantage, albeit at a cost:

- Add more overlay nodes at non-stub networks, so that good non-violating overlay paths can be created, as noted from our observations in Section violations:classification.
- Negotiate deals with ASes traversed by violating shortest overlay paths (transit and exit), so that overlay traffic is allowed to pass through. This in essence creates an overlay policy that supplements the native policy, but is independently managed. This is the only option available to mitigate exit policy violations because enforcing all exit policies will cause the overlay traffic to take the direct native route.

In this section, we consider a generalized approach where the two schemes above are deployed individually or in combination. In this general scheme, new nodes may be deployed in some parts of the network to create non-violating paths, while ASes in other parts of the network are paid to allow violations (of both transit policy and exit policy). One can also think of the two schemes used simultaneously: an overlay node is added to create some good violating paths and the ASes are paid to allow these paths. By adopting these two approaches, we obtain overlay paths that are better than what is achieved when all ASes perform policy-aware filtering.

This solution allows the OSP to share the cost originally incurred by the native network in return for obtaining a routing performance advantage. Hence, we refer to it as the *cost-sharing* approach. It is conceivable that such an approach can be adopted to relax any objection raised by the native network, thereby fostering a higher level of economic cooperation between the two layers. Adopting this cost-sharing approach is crucial to put an end to the selfish conflict between the two layers, which often leads to a deterioration in routing performance[100].

The basic idea of the cost-sharing strategy is to monetize the objections of the violated ASes i.e., we determine the amount an OSP needs to pay the native layer, so as to use an objectionable inter-domain link or intermediate AS for its traffic. It could be based on the monetary loss incurred by the native service provider for sending traffic in a different direction. Say an OSP pays C_1 to its host AS X for certain usage and causes the host AS to incur an higher expense of C_2 when it performs relaying, then the OSP can pay the overhead of $C_2 - C_1$ to the host AS.

The cost-sharing solution involves the following three costs that are paid by the OSP to the native network, *over the lifetime of the overlay*[4]. Each of these fees can be zero if the overlay nodes are already permitted by the end-user agreement to adopt any overlay paths:

- New node fee, N_i: Cost for adding a new overlay node in native AS i and for the associated network resource usage.
- Transit permit fee, T_i: Cost for making a native AS i allow the transit-violating overlay traffic to be relayed through its network.
- Exit permit fee, E_i: Cost for making a native AS i allow the exit-violating overlay traffic to exit its network.

Note that it is possible the same AS experiences transit violations in one set of multi-hop overlay paths and exit violations in a different set of multi-hop overlay paths. Nevertheless, we devised it such that the OSP pays individual permit fees for better clarity of our

[4]We avoid usage-based billing to remove effects of traffic variability.

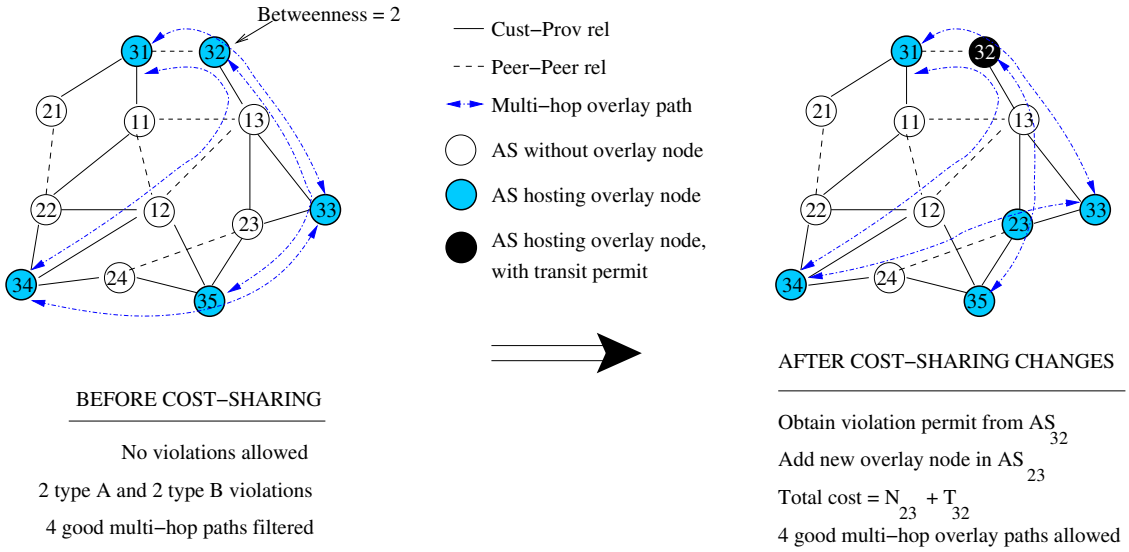


Figure 21: Illustration of the cost-sharing approach, where we pick the optimal set of ASes to use for relaying. In the above figure, the AS numbers indicate which AS is the provider and which is the customer. For instance, an AS with AS number 12 will be a provider of AS with AS number 22.

approach.

Fig. 21 illustrates the cost-sharing approach in a typical overlay network spread over multiple native ASes. The figure shows the transition from an original network with four violating overlay paths to a network where these paths have been "legalized", by adding a new node to AS_{23} and obtaining a transit permit from AS_{32}. By making these purchases, the overlay service provider obtained 4 multi-hop overlay paths with good performance. Moreover, there were no new violations caused when the new overlay node was added into the overlay topology because AS_{23} is a non-stub domain. Consider for example the route between the overlay node in AS_{34} and that in AS_{33}. The shortest overlay path (AS_{34}-AS_{24}-AS_{35}-AS_{23}-AS_{33}) constitutes a violation of type A and is disallowed by the native layer. This causes the overlay route to adopt the direct native route AS_{34}-AS_{12}-AS_{13}-AS_{33}, which is substantially longer. However, the cost-sharing approach helped obtain a better route through the new node in AS_{23}. This leads to the new overlay route AS_{34}-AS_{24}-AS_{23}-AS_{34} and a corresponding path gain.

However, neither option eliminates the exit policy violation experienced by AS_{34}. The

OSP has fewer alternatives for legitimizing the exit policy violations. This is because any deviation from the direct native route, with an objective of attaining a higher routing gain, will cause more exit violations. Hence, the OSP is required to compensate AS_{34} for the exit policy violation in order to retain the routing advantage.

4.6.1 Deploying the Cost Sharing Scheme

We now consider the question of how to best deploy the cost-sharing scheme described above. The problem we address is the following: *Given a certain budget, new node costs and permit costs, how should an overlay service provider determine where to position new nodes and what permits to obtain, in order to maximize its performance advantage within the constraints of the budget.*

Based on this problem statement, we can see that the solution to the cost-sharing approach is comprised of two components:

$$\mathcal{N} = \text{Set of ASes where new nodes are placed}$$

$$\mathcal{T} = \text{Set of ASes being paid for transit permits}$$

$$\mathcal{E} = \text{Set of ASes being paid for exit permits}$$

We represent the overall solution as $\mathcal{S}$, where $\mathcal{S} = \{\mathcal{N}, \mathcal{T}, \mathcal{E}\}$. A solution yields a new set of shortest paths in the overlay, $\mathbb{H}$. This set is made up of a set of non-violating or permitted paths, $\mathbb{H}_N$ and a set of violating paths, $\mathbb{H}_V$. The overlay paths in $\mathbb{H}_N$ can provide a gain in routing performance over native routing (as described in Section 4.4.2). However, the violating paths in $\mathbb{H}_V$ cannot be used (because of the assumption that these are filtered) and the overlay resorts to using the single-hop overlay link. This provides no advantage to overlay routing. An ideal solution, hence, is where $\mathbb{H}_V = \emptyset$.

The cost-sharing deployment problem can be formulated as:

$$\max_{\mathcal{S}} \ \text{Gain}(\mathcal{S}) \text{ such that } \text{Cost}(\mathcal{S}) \leq B, \text{ where}$$

$$B = \text{Budget allocated}$$

$$\text{Cost}(\mathcal{S}) = \sum_{i \in \mathcal{N}} N_i + \sum_{j \in \mathcal{T}} T_j + \sum_{k \in \mathcal{E}} E_k$$

$$\text{Gain}(\mathcal{S}) = \frac{\sum_{i \in \mathbb{H}_\mathbb{N}} \text{Gain}(i)}{|\mathbb{H}|}$$

To solve the cost-sharing problem in the context of inter-domain policy violations and policy-aware filtering, we need the following details about the native network and the original overlay network[5]:

- Overlay network topology (node location, link connectivity).
- Estimated length of each overlay link, based on the metric of choice.
- The AS-level path of each overlay link and the relationships between each pair of AS present in the AS-level path. This helps us determine which relaying operations are filtered by the native layer.
- The hypothetical shortest overlay path computed without concern for inter-domain violations. We denote these paths as $\mathbb{H}'$. They represent the highest achievable gain. These are the routes we characterized in Section 4.4.
- The costs involved in adding nodes and for obtaining permits from ASes, computed from various native-overlay business agreements.

The cost-sharing problem is complicated because of the policy constraints that need to be satisfied. Moreover, the gain value is non-additive with respect to the different ASes used for relaying, i.e., if we know the gain $G1$ achieved when only $AS1$ is paid money for relaying and the gain $G2$ achieved when only $AS2$ is paid money for relaying, we cannot say that the gain achieved when both $AS1$ and $AS2$ are used for relaying is $(G1 + G2)$. This makes our problem different compared to other conventional weight-constrained shortest path problems[72, 30, 33].

Obtaining an optimal solution $\mathcal{S}$ is a hard problem[6]. Hence, we use insights from our analysis in Section 4.4 and 4.5 to derive greedy heuristics to obtain a reasonable solution.

[5]Most of these can be obtained by a procedure similar to that we adopted in Section 4.4.

[6]The cost-sharing problem can easily be shown to be NP-hard by performing a reduction to the set-cover problem, which is known to be NP-complete[60].

- Set $\mathcal{N} = \mathcal{T} = \mathcal{E} = \emptyset$
- For each path i in $\mathbb{H}'$
  - For each relay node j in path i, betweenness(j)++
  - Compute Gain(i)
- Sort all overlay nodes in decreasing order of $\frac{\text{betweenness}}{T \text{ for host AS of node}}$
- while total_cost $< B_{th}$
  - j = Get Next Entry(Sorted list of overlay nodes)
  - k = Host AS of node j
  - $\mathcal{T} = \mathcal{T} \cup k$; Obtain transit permit from host AS k
  - Compute potential overlay paths $\mathbb{H}$ after obtaining permit
  - In $\mathbb{H}$, determine AS l with highest $\frac{\text{number of exit violations}}{E \text{ for that AS}}$ value
  - $\mathcal{E} = \mathcal{E} \cup l$; Obtain exit permit from AS l
  - total_cost = total_cost + T_k + E_l
- Sort all overlay paths in $\mathbb{H}'$ with violation A/B in decreasing order of $\frac{\text{path gain}}{N \text{ for upstream provider AS}}$
- while total_cost $< B$
  - i = Get Next Entry(Sorted list of overlay paths)
  - k = Upstream provider AS in path i
  - If AS $k \in \mathcal{N}$, continue
  - If there exists no path between AS k and destination of path i, continue
  - $\mathcal{N} = \mathcal{N} \cup k$; Add new node to provider AS k
  - Compute potential overlay paths $\mathbb{H}$ after obtaining permit
  - In $\mathbb{H}$, determine AS l with highest $\frac{\text{number of exit violations}}{E \text{ for that AS}}$ value
  - $\mathcal{E} = \mathcal{E} \cup l$; Obtain exit permit from AS l
  - total_cost = total_cost + N_k + E_l
- Solution $\mathcal{S} = \{\mathcal{N}, \mathcal{T}, \mathcal{E}\}$

Figure 22: Greedy scheme for the cost-sharing problem.

4.6.2 Greedy Heuristic Solution

Our heuristic solution is shown in Fig. 22. It produces the solution in two phases. In the first, it obtains transit permits for violating paths in a particular order, until the cost of buying permits exceeds a threshold value B_{th}, which is less than or equal to the total budget B. In the second phase, the remaining budget (if any) is used to add new nodes to provide more non-violating paths. The ordering of the two phases is motivated later. During each phase, we simultaneously resolve any exit violations that arise. The order of compensation of the individual ASes in each phase is motivated by the following insights that we obtained from our previous analysis:

1. We observed in Section 4.4 that most of the violations are in the form of a transit to an upstream provider (Type A and B). Hence, it is desirable to add overlay nodes at these upstream providers (relative to the point of violation), so as to bypass the overlay node associated with the violation. Our heuristic, therefore, adds overlay nodes to intermediate ASes in the unconstrained shortest overlay paths $\mathbb{H}'$, starting with the violating overlay paths which achieve the highest gain. If there exists an upstream provider in a violating path with very low new node fee N, then it would be in our best interest to give a higher preference to such placing a node there. We achieve this by normalizing the value of path gain by the new node fee for the upstream provider (unless the cost is zero, for which we just use the absolute value), as done in the approximation algorithm for the set-cover problem[73].

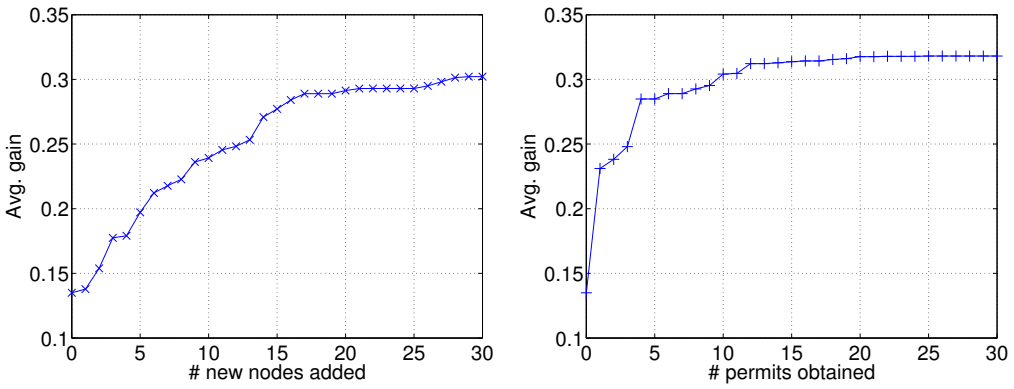
2. From the results in Section 4.4, we know that most of the violations are committed at stub ASes hosting overlay nodes. Moreover, betweenness plots in Section 4.4 indicated that there are a few overlay nodes that are key to most of the overlay paths. Hence, by merging both observations, we negotiate deals with the stub ASes, in the decreasing order of relay betweenness in $\mathbb{H}'$, to permit the violating overlay traffic to be relayed through. Similar to the previous discussion, we normalize the betweenness value of a particular overlay node by the permit fee for the corresponding host AS.

Adding new nodes or permitting a transit typically has the tendency of changing the adopted overlay path, and thereby the exit points. This indicates that the exit violations must be resolved *after each cost-sharing move*. Our analysis of the location of the exit violations showed no evidence that a particular AS is always violated when a certain overlay node is used in that link. However, we did observe a higher correlation between the location of the exit violation and the intermediate relay used. This further corroborates our choice of estimating exit violations after determining the exact intermediate relay to use at each step.

Based on our observations from Fig. 17, we posit that obtaining exit permit from the AS experiencing the most exit violations is the best choice. Nevertheless, if there exists an exit violated AS with very low exit permit fee E, then it would be in our best interest to give a higher preference to obtaining an exit permit from it. We achieve this by normalizing the number of exit violations at each AS by its exit permit fee (unless the cost is zero, for which we just use the absolute value), as done in the approximation algorithm for the set-cover problem[73]. This provides a good tradeoff between budget and path gain.

The exact value we adopt for the budget threshold B_{th} depends on the actual topology of the native and overlay network. When threshold B_{th} = budget B, we only obtain permits to improve overall gain. When threshold $B_{th} = 0$, we only add new nodes to improve overall gain. This shows that the threshold value B_{th} has a direct influence on the effectiveness of the cost-sharing approach, by controlling the decision of which heuristic to follow. Generally, the ideal threshold value can be found from the betweenness plot in Fig. 15, which shows that only a few nodes are repeatedly present in many overlay paths. Hence, we can look for a knee point in the betweenness curve to determine the appropriate threshold value. Based on the betweenness values observed in our case study and in other simulated overlay networks, we recommend the following rule of thumb for setting the threshold value:

$$B_{th} \approx (\#\text{ relays with betweenness} > \frac{\text{max. betweenness}}{2}) \times P$$



(a) Only add new nodes

(b) Only obtain transit permits

Figure 23: Average gain achieved with the two individual heuristics.

4.6.3 Applying the Heuristic to Our Case Study, assuming no exit policy restrictions

In this subsection, we show results from applying our heuristic to the overlay case study analyzed previously. In these results, we assume that the permit fee is the same for all ASes ($T_i = P \ \forall$ AS i) and the new node fee is the same for all ASes ($N_i = N \ \forall$ AS i). In Fig. 23(a) and 23(b), we first show the effect of adding nodes or obtaining transit permits in the order specified by the individual heuristic. We make two observations about the heuristics. First, as expected, the ordering of new nodes to add and transit permits to obtain give the desired effect of producing the best gain in the first few nodes added or transit permits obtained. Second, we observe that the gain from the initial few transit permits can be substantial. Hence our decision to obtain transit permits first and then add nodes in the greedy algorithm of Fig. 22.

We next applied the cost-sharing algorithm for different values of the threshold value, while keeping the overall budget B at a constant value of $20 \times P$. We plot, in Fig. 24, the solutions obtained for different ratios of N/P. We can see that the knee of each curve lies around a threshold value of 5 or 6. This is coherent with our rule of thumb. We observed in Fig. 23(b) that the gain achieved by obtaining permits saturates after a certain point. Hence, having a high threshold value and filling up the budget with permit expenses is not desirable because we lose on any potential gain that can be achieved by adding new nodes.

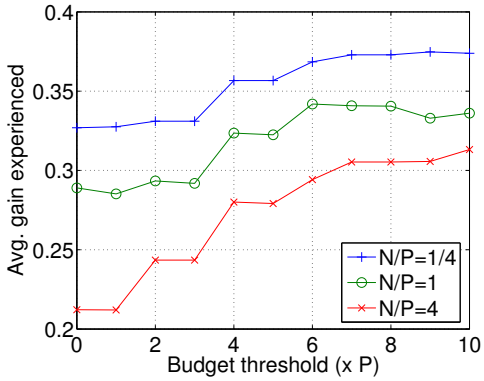
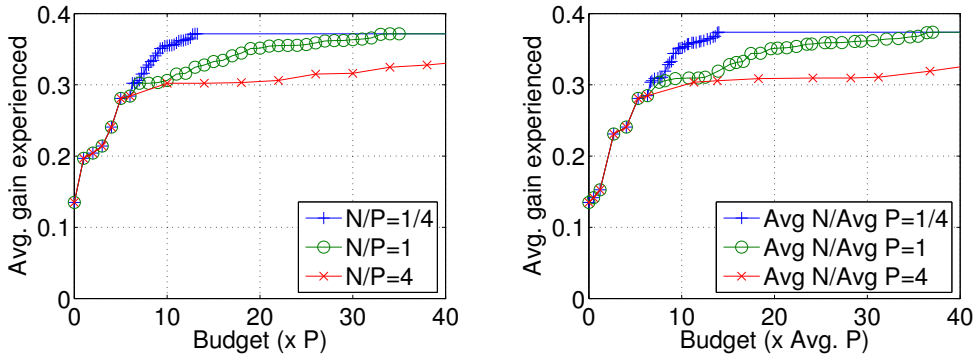


Figure 24: Solutions for different values of B_{th}, when budget $B = 20 \times P$

Keeping this in mind, we varied the threshold value only between zero and $B/2$.

We next consider the effect of the total budget B on the achievable performance. Fig. 25(a) shows the performance when we apply our greedy algorithm for $B_{th} = 6 \times P$, with varying budgets. The ratio between the new node fee and the permit fee determines how effectively the remaining budget will be utilized after obtaining sufficient permits. Therefore, it has a direct bearing on the achieved gain. As expected, for a fixed budget threshold, the higher the N/P ratio, the lower the gain achieved. By comparing the plots in Fig. 23 and Fig. 25(a), we observe two important points - i) the greedy algorithm performs better than the individual heuristics for each value of the budget, ii) the highest gain achieved in the cost-sharing scheme is greater than what is achieved in $\mathbb{H}'$ (which is the case where all ASes permit violations). These two observations corroborate our combined cost-sharing approach.

All previous experiments assumed equal N_i and T_i. For the same overlay case study, we computed the solution $\mathcal{S}$ for a random distribution of the costs N_i and T_i. The new node fee was uniformly distributed between $[0.5\times N,\ 1.5\times N]$ and the permit fees between $[0.5\times P,\ 1.5\times P]$, thus maintaining the average values at N and P. Fig. 25(b) presents the gain achieved in this scenario. We observe that the gain achieved for a certain budget is comparable with the earlier simplified scenario. This shows that the algorithm is more influenced by the average costs, rather than the absolute value.



(a) When P is equal ∀ ASes and N is equal ∀ ASes.

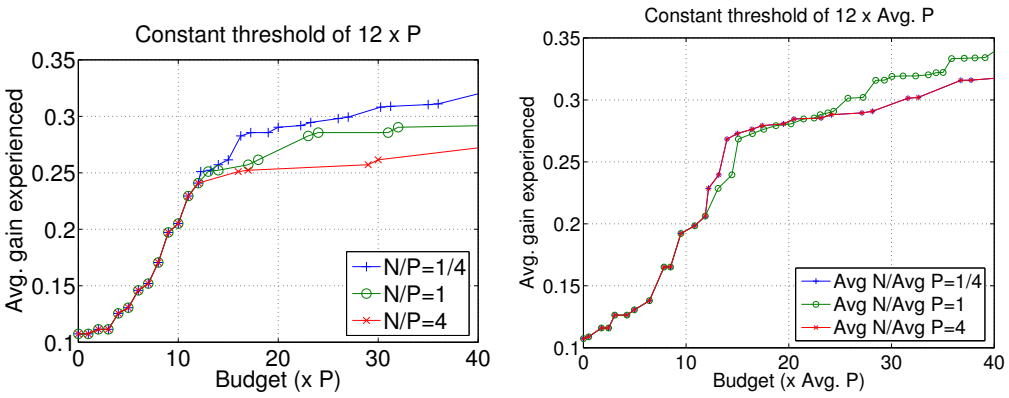
(b) When P and N for each AS is uniformly distributed.

Figure 25: Solutions with the cost-sharing greedy scheme, when $B_{th} = 6 \times P$

4.6.4 Applying the Heuristic to Our Case Study, with exit policy restrictions

We now address the exit policy restrictions. As seen from our greedy algorithm in Fig. 22, the violations are to be resolved simultaneously. For better understanding of the results, we assume the transit permit fee T_i and the exit permit fee E_i for a particular AS i to be equal to P, and the new node fee N_i to be equal to N. Fig. 26(a) presents the gain observed for each expenditure by the OSP, when the budget threshold was configured at $12 \times P$. From the figure, we observe that the OSP is able to achieve a significant improvement in routing performance that is commensurate with the budget spent. We also see that the increase in gain is sluggish when the budget is low. This is because all ASes in the system filter out violating traffic initially. Obtaining transit permits in that scenario does not cause much change in the gain, until a few critical exit violated ASes are compensated. After crossing that point, the achieved gain increases rapidly with the increase in budget.

Further, we conducted the cost-sharing experiment for a random distribution of the costs N_i, T_i and E_i. The new node fee was uniformly distributed between $[0.5\times N,\ 1.5\times N]$ and the permit fees between $[0.5\times P,\ 1.5\times P]$, thus maintaining the average values at N and P. In this scenario, the improvement achieved for a certain budget was similar to the earlier simplified scenario in Fig. 26(b), showing that the algorithm is more influenced by average costs here as well.



(a) When T and E is equal ∀ ASes, and N is equal ∀ ASes.

(b) When T, E and N for each AS is uniformly distributed.

Figure 26: Solutions with the cost-sharing greedy scheme with both transit and exit policy restrictions, when $B_{th} = 12 \times P$

Though there exists no unique exit violated AS that needs to be appeased after each cost-sharing step, our greedy sequential approach offers a reasonable improvement in routing performance for the OSP. Moreover, we do not notice a case where a cost-sharing step fails to make a difference because the corresponding exit violated AS is not compensated.

It is conceivable that this cost-sharing procedure be incorporated from the initial stages of the overlay topology design, rather than serving to supplement an existing overlay topology. This design approach is a hard problem with two unconstrained objectives: 1) Achieving good overall path gain, 2) Resolving arising policy violations. The problem poses different set of challenges as the $\mathbb{H}'$ (set of shortest overlay path without concern for policy violations) does not exist initially and grows with the size of the overlay topology. We reserve further investigation of this problem for future study.

4.6.5 Network Characteristics

Our cost-sharing approach improves on a given scenario, without creating any new policy violations, by exploiting the following three properties:

1. The property that there exists non-stub ASes that can offer a good route to a destination.

2. The betweenness property of overlay nodes, wherein there exists a small set of overlay nodes that are present in many overlay paths.

3. Exit violations are distributed in a non-uniform manner across multiple ASes.

We understand that many of the conclusions drawn in this section may seem limited by the fact that they originate from a single Planetlab dataset. In this subsection, we establish the generality of our approach by showing that these three properties hold true in a wide variety of networks.

The first property can easily be reasoned to be true based on the knowledge that inter-domain routing is policy-constrained and does not always adopt the shortest route to a destination. By adding an overlay node at the upstream provider, we are able to force the AS to adopt the shorter route, thereby regaining the routing advantage.

To verify the second and third property, we simulated 90 random overlay networks with varying number of overlay nodes in stub ASes and varying out-degree (extent of multihoming) of the host AS. In particular, the number of stub ASes was set at 35, 40 or 45, and the maximum out-degree of the host AS was bound to 10, 25 or 40. This gives a total of 9 combinations, each of which was simulated 10 times.

In each run, we picked 50 host ASes, catering to the above mentioned characteristics, from a list of 21,416 ASes observed in the different BGP route dumps used in Section 4.4. We computed the AS-level route of each overlay link by using the BGP routes collected from multiple vantage points as input and performing policy routing between the two host ASes[7]. In addition, we assigned random latency values for each inter-AS link (in the range of 10-50 ms) and computed the shortest overlay path between each pair of host AS. When we inspected the betweenness of each overlay node, we observed that the betweenness property indeed holds in all scenarios.

Further, we applied our cost-sharing algorithm to each of the 9 scenarios and plotted the results in Fig. 27. We assumed no exit policy restrictions for this experiment. The gain achieved was averaged over the 10 different overlay networks generated for each scenario.

[7]We computed the shortest AS-path that does not violate native layer policy. This is an approximation as the actual routing tables and the policies are unavailable.

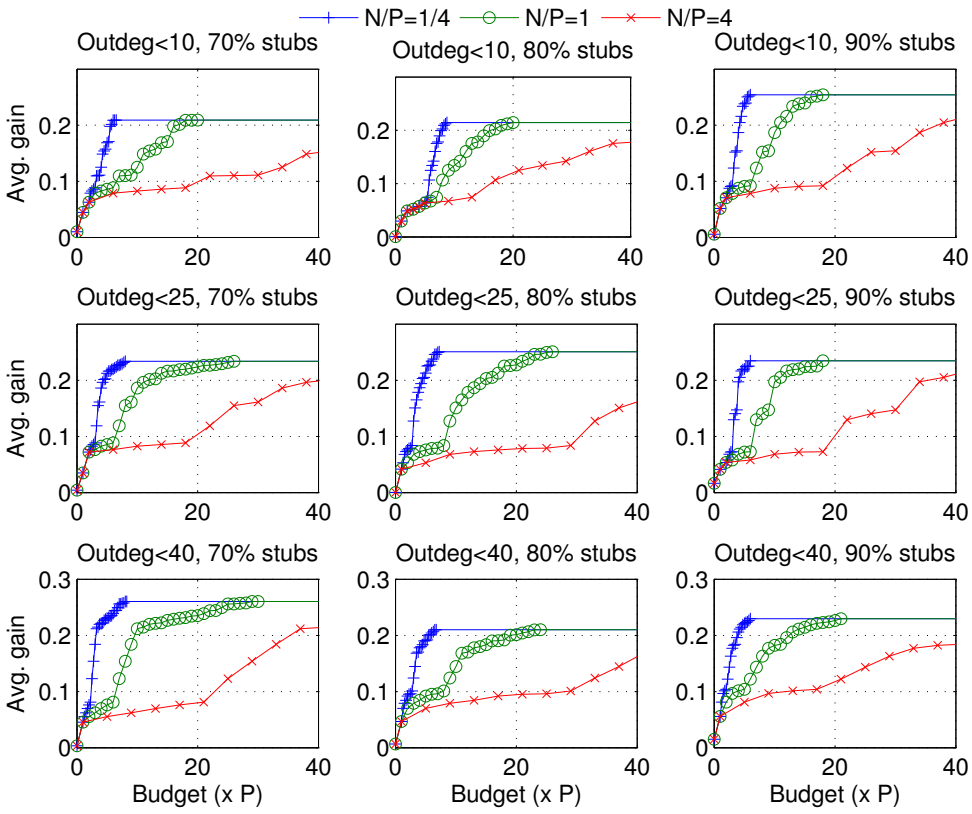


Figure 27: Cost-sharing solutions for the 9 simulated scenarios

In all scenarios, we notice a sharp increase in the gain when the budget is low. After some point, though, adding more budget does not lead to significant gain improvement. As the initial permits obtained provided a higher gain than what the new node addition provided, our choice of B_{th} (according to the rule of thumb) in each run is justified. The individual plots in Fig. 27 are similar to those observed in our case study, indicating that our approach can be used to improve performance of many possible overlay topologies.

4.7 Related Work

Our work classifies violations according to results in [102], which tries to understand the BGP misconfigurations that are prevalent in the current Internet. With advances in BGP measurement studies and data sources[59, 103, 131, 128, 166, 172, 99, 75, 166], it is now possible to determine the different AS policies endured by a packet exchanged between two end systems. We combine both these directions of work in the context of overlay routing to analyze violations of overlay traffic.

The second motivation of our work lies in the rapid development in the market for traffic management products[21, 134, 161, 117, 144], based on ASes' need to control the influx of overlay traffic. To identify and filter these overlay packets, most products adopt a flow-signature-based approach[150] that develops some form of correlation between the incoming and outgoing packets, or a communication-pattern-based approach[81]. However, there is very little understanding of the impact of filtering on the user experience. We address this issue in our thesis.

4.8 Summary

In this chapter, we investigate the concern that overlay routing derives performance advantages by violating native routing policies. Specifically, we investigated violations of the transit policy and exit policy, caused by multi-hop overlay paths. As more overlay applications are introduced to subvert the functionality limitations of the Internet, the frequency of policy violations can become substantial, which would increase the relevance of this work. Further, we analyze the impact of native layer traffic filtering attempting to prevent these violations. We showed that a clear tradeoff exists between the number of policy violations

and the penalty incurred when the native layer enforces these policies. It is conceivable that more networks will start filtering overlay traffic. We showed that even policy-aware filtering can be detrimental to the overlay routing efficiency, while blind filtering can completely remove any incentive to use overlay routing. In this context, we propose a cost-sharing approach that allows the overlay service provider to recover the overlay routing advantage through payments to native network operators. Further, we prescribed a heuristic-based algorithm for solving the cost-sharing problem with a certain budget. We believe that this approach provides a framework to legitimize snative policy violations and allow the benefits obtained by the overlay to be directly related to costs incurred by the overlay service provider.

CHAPTER V

INTERACTION BETWEEN OVERLAY ROUTING AND TRAFFIC ENGINEERING

5.1 Introduction

In a resource-constrained world, where the native layer performs traffic engineering (TE), the selfish behavior of overlay routing tends to backfire and causes all traffic to suffer route oscillations, increased routing cost and resource starvation[100, 98, 83]. Further, instability and sub-optimality is exacerbated when there is a conflict in objective between the two entities, viz. overlay routing that aims to minimize the latency between the nodes of a service overlay network[39], and traffic engineering that aims to balance the load in the underlying native network[112, 55]. Although this non-cooperative interaction is a well-studied problem, past research does not suggest ways to avoid associated shortcomings and attain an optimal operating point.

In this context, *our goal is to propose strategies that obtain the best possible performance for a particular layer by predicting or counteracting the other layer's reaction, while steering the system towards a stable state.* This is similar to the Stackelberg approach where one player acts as a leader and the other players are selfish followers[87]. We refer to the layer that makes the first unconventional route adjustment as the *leader* and the layer that reacts to this change as the *follower*. As these strategies allow one layer to firmly assert its performance, without any future deterioration, we refer to them as *preemptive* strategies. The general idea is to insure that the leader picks those optimal routes for which the follower has no other alternative or volition but to retain the same routes. Specifically, we propose preemptive strategies for the following two scenarios:

1. When overlay applications can estimate the characteristics of the underlying native network and can sufficiently predict its behavior for a certain load distribution. In the context of service overlay networks, the objective of the leader is to *minimize the*

end-to-end latency of the overlay paths.

2. When the native network is aware of the selfish overlay network and can sufficiently predict its behavior for a certain network topology. In the context of traffic engineering, the objective of the leader is either to *minimize the maximum link utilization*, or to *minimize the overall network cost.*

Unfortunately, prediction in the true sense is not a pragmatic solution owing to three main issues. Firstly, overlay networks and native networks maintain independent routing tables and have different network span, making it unrealistic to procure complete knowledge about the other layer's functioning. Secondly, the prediction process makes it essential to contrive a relation between the latency objective and the load balancing objective, which does not exist in reality. Lastly, determining the exact routes to be prescribed by each layer, even in the presence of complete information, is a hard problem[173].

We work around these limitations by profiling the multi-layer interaction, and propose simple strategies for the leader to proactively prepare itself for the follower's reaction (response). As this represents a *repeated game*[57], where the players have continuous sequential interaction, it is possible to capitalize on historical observations and to gradually learn the desired action. Specifically, we propose two classes of strategies – *friendly* or *hostile* – for each layer.

In the friendly strategy, one layer picks routes in such a manner that it improves its performance without defeating the objective of the other layer. The fundamental idea behind the friendly design being that the follower does not get instigated to react if the leader operates within certain bounds acceptable to the follower. On the other hand, a hostile strategy improves the performance of one layer primarily by defeating the objective of the other layer, with minimal chance for recuperation. The fundamental idea behind the hostile design being that the leader can cause irrecoverable problems for the follower in an effort to leave the follower no viable option to react.

As overlay applications proliferate, it is highly likely that the amount of selfish overlay traffic will experience significant growth in the future. Moreover, network virtualization

through overlay networking is seen by many as a potential future for the Internet[141]. This tends to alarm the current ISPs about the impending destabilization of its network. In such a context, deploying our strategies in either layer is crucial to eliminate the instability (persistent route oscillations) generally observed in the non-cooperative interaction[98, 83, 138], without compromising on route optimality. Though we do not recommend the hostile strategies, we propose and analyze them in this thesis to prepare each layer for possible hostile attacks from the other layer in the future.

Our contributions are three-fold:

1. We provide an understanding of the objective conflict in the multi-layer interaction and its detrimental effects.

2. We develop means to mitigate the inherent instability in the system without compromising on the routing performance.

3. We propose simple, yet efficient, strategies that help a particular layer achieve near-optimal performance, with limited information of the other layer.

The remainder of this chapter is organized as follows. We briefly describe related work in Section 5.7. We present the issues involved in the interaction and the model for evaluation in Section 5.2. We characterize the behavior of the multi-layer interaction in Section 5.3 by a simulation study. Sections 5.4 and 5.5 propose preemptive strategies that improve the performance of the overlay layer and native layer respectively. Section 5.6 presents the performance achieved when both layers adopt a preemptive strategy. We summarize this chapter in Section 5.8.

5.2 Performance Evaluation

In this section, we describe the exact behavior model of each layer and present our methodology of performance evaluation.

5.2.1 Network Model

We investigate the interaction between the following two entities:

5.2.1.1 Traffic Engineering

TE is a crucial procedure in modern ISP networks to balance load and remove bottlenecks. Typically, it uses a particular snapshot of the traffic demand matrix to derive the set of routes that achieve a specific load-balancing objective. The frequency of re-engineering the routes depends on the amplitude of change in the traffic matrix or the desired periodicity. In our work, we study the effects of adopting one of the following two objectives:

- Minimize the overall cost of the network (as proposed by [55]), where the cost $\phi(a)$ of an individual link a is modeled using a piecewise-linear, increasing, convex function.
- Minimize the maximum link utilization in the network (as used by [79, 162]), where the utilization of an individual link a is defined as the ratio between the cumulative load X_a in the link and the capacity C_a of the link.

TE can be implemented by means of MPLS[35], where the traffic between two nodes is partitioned and transported over one or more pre-configured tunnels, or OSPF/ISIS[55], where the IGP link metrics are optimized to approximate the solution of MPLS-TE. As MPLS achieves the optimal TE objective, we only focus on the interaction between overlay routing and MPLS-TE.

We model the native network as a directed graph $G = (V, E)$, where V is the set of nodes and E is the set of directed links with finite capacity constraints. The latency of each physical link is the sum of the propagation delay and the queuing delay. We analyze two different cases in the rest of the chapter: one where the queuing delay is negligible in comparison to the propagation delay and one where it is non-negligible.

5.2.1.2 Overlay Routing

We focus on a service overlay network, which is managed by a single operator, and which offers latency-optimized paths to actual end systems. To achieve this, the overlay layer maintains a routing table that is independent of the underlying network[6, 51, 155] and deploys some form of dynamic routing to adapt to changing conditions in the native network. Following standard terminology, an *overlay link* represents the direct native route between

two overlay nodes, which in turn comprises of one or more native links, and an *overlay path* is made up of one or more overlay (virtual) links. This overlay path represents the end-to-end route taken by the application traffic.

We model the overlay network as a directed graph $G' = (V', E')$, with V' being the set of nodes and E' being the set of edges. We assume that the overlay topology is given and aim to improve over current performance. We assume complete mesh connectivity of overlay links between the overlay nodes. The overlay network periodically monitors the state of the overlay links and the latency incurred by each of them. Based on the collected data, the overlay performs some form of link state routing to optimize its objective.

5.2.2 Interaction between the Layers

Each entity described above operates solely on the results of the network monitoring process and is otherwise oblivious to the dynamics of the other layer. This independent operation of routing protocols in the two layers, as seen in today's networks, has the following two inherent drawbacks:

- Misalignment of objectives: In the attempt of deriving shortest paths, overlay routing tends to reuse a short overlay link in multiple paths[1]. This causes the load on that native route to escalate beyond the expected demand, thereby upsetting traffic engineering. Similarly, in an effort to balance load, TE may offer native routes that span under-utilized routes in remote regions of the network, causing a stretch in the overlay link latency. This shows a serious misalignment in objectives leading to contention and associated route oscillations.

- Misdirection of traffic matrix estimation: The TE operation used by the ISPs relies heavily on the estimate of the traffic matrix. In the case of overlay networks, irrespective of the TE protocol or objective, the drawback is that the estimated traffic matrix is not reflective of the actual end-to-end source-destination demand[100, 83]. Hence, there is a certain amount of misdirection in the load-based optimization procedure of

[1]It has been shown on a Planetlab testbed that the popularity (betweenness) of certain nodes in the overlay network is non-uniformly distributed[137].

TE. For instance, consider the traffic on two overlay paths $A-B$ and $A-B-C$. The traffic on the overlay link AB cannot be differentiated based on the true destination. This makes the load distribution process rigid.

The interaction between overlay routing and traffic engineering is non-cooperative and recurring. Each layer optimizes the routes to suit its local objective in succession. We refer to the duration between two iterations of TE as a *round*. The number of overlay routing operations between two TE operations may vary based on the probing frequency.

Fig. 28 illustrates this multi-layer interaction by means of a simple example. In the figure, the numbers indicate the available bandwidth on each native link and the latency value of each link in ms. We do not show the IGP metrics. We assume that the traffic is split evenly between available equal cost multipaths and the latency value of each native link is a constant parameter. The overlay traffic matrix contains 1Mbps of traffic from each node to every other node. We notice that any reconfiguration in one layer's routes leads to a substantial change in the other layer's state (link load profile in the case of TE or link latency profile in the case of overlay routing). Such a system takes longer to stabilize in the presence of resource constraints. The system in the example takes longer than 10 rounds to reach steady state.

5.2.3 Performance metrics

Based on the particular TE objective, the routing performance of the native layer can be measured by one of the following two variants:

- *Native cost*, in the event the ISP chooses to minimize the overall cost incurred by its network. The native cost is computed as $\sum_{a \in E} \phi(a)$, where a represents a link in the set of edges E and ϕ is the summation of the piecewise integral of the cost increase function.

- *Maximum utilization*, if the objective of the ISP is to minimize the maximum link utilization observed in its network. The maximum utilization is computed as $\max_{a \in E} \frac{X(a)}{C(a)}$, where a represents a link in the set of edges E.

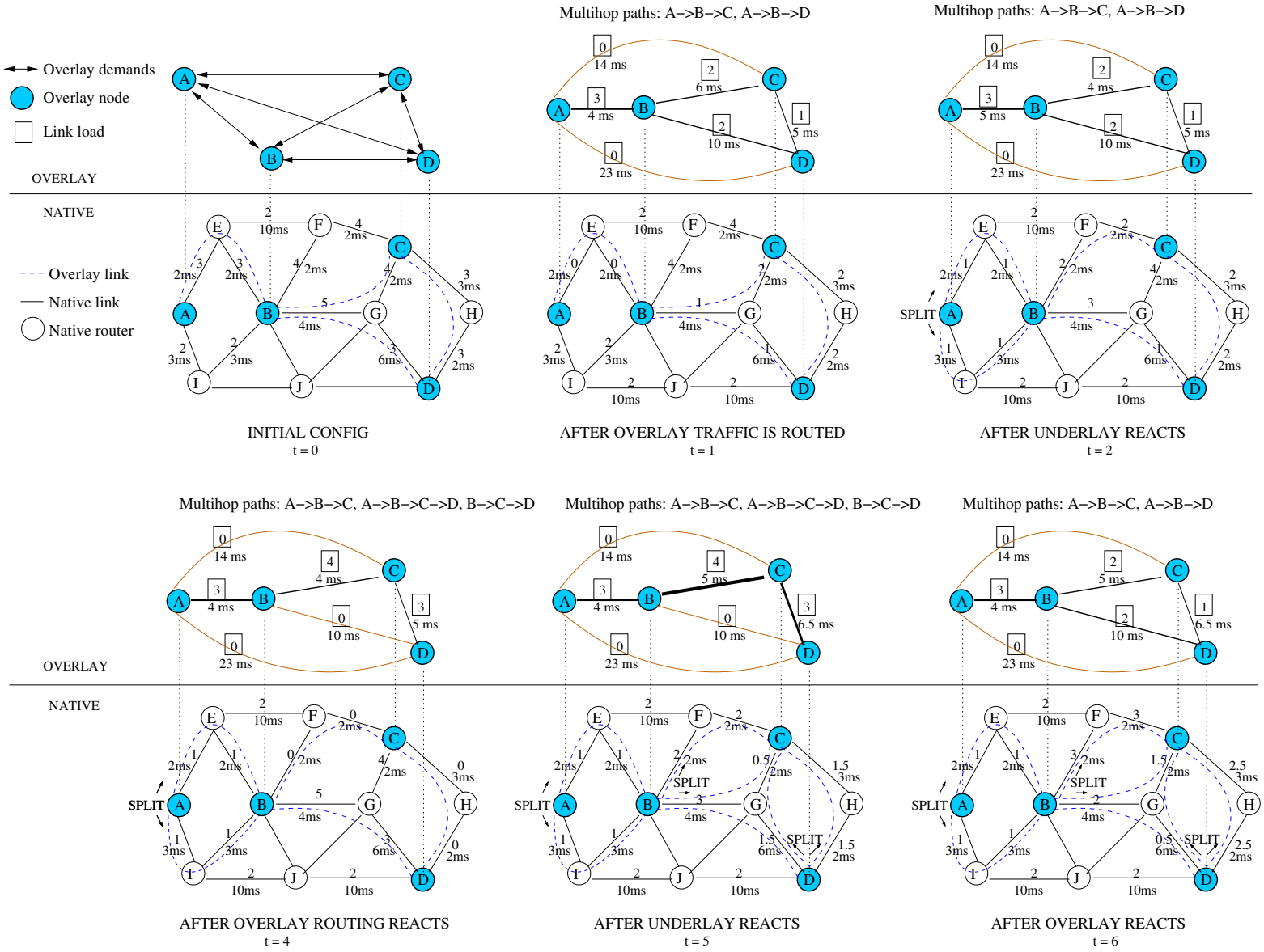


Figure 28: Illustration of 2 rounds of interaction between overlay routing and TE.

In our work, we focus on service overlays that offer lowest latency routing service. Hence, the routing performance of the overlay layer can be measured by:

- *Average latency*, which is defined as the average of the end-to-end latencies observed across all overlay paths with non-zero traffic demand.

When there exists a conflict in the objective between the two layers, the system tends to become unstable, leading to frequent alterations in the route taken by existing traffic. These changes in route can happen to all flows at the end of traffic engineering, or just to overlay flows at the end of overlay routing. Each such route change is referred to as a *route flap*. Route flaps can be a serious problem in case of TCP, VoIP and other traffic that relies on packet ordering and is sensitive to jitter[17]. As a result, the end-to-end performance is hurt. Moreover, the route flaps serve as an important indication of the instability in the system. Hence, the next performance metric of interest is:

- *Number of route flaps*, which is the sum of route changes observed in existing flows after a routing operation.

During the multi-layer interaction of overlay routing and TE, there are operating points where the performance of a particular layer is the best it can be. We refer to this performance as the *best-case*, or *optimal*, performance of that layer. This best-case performance can be computed as the minimum of the objective value attained in any of the rounds. However, that layer is usually unable to retain this best-case performance, as the other layer annuls it during its routing operations.

The preemptive strategies we propose attempt to steer the system towards the best-case performance of the leader. However, as shown later, the leader is not always able to achieve the best-case performance, and sometimes incurs a minor loss of routing performance, as a tradeoff against the gain in system stability. We use the following performance metric to estimate the effectiveness of the preemptive strategies we propose:

- *Inflation factor*, which is defined for each layer as:

$$\text{Inflation factor} = \frac{\text{Steady state obj value with strategy}}{\min_{t=0..\infty} \text{Obj value without strategy}}$$

In the case of the leader, this factor is used to determine if the leader achieved the best-case performance at the steady state, or if the leader had to accept a minor tradeoff to retain its stable performance. In the case of the follower, this factor reflects the amount of sub-optimality incurred by the follower, due to the leader's preemptive action.

5.2.4 Simulation Setup

In this subsection, we describe the simulation setup that we used to evaluate the performance of the multi-layer interaction (with or without the preemptive strategies). Clearly, the routing performance in this multi-layer scenario is topology-specific and load-specific. Hence, we simulate multiple overlay topologies over multiple native networks, with varying levels of traffic, to improve the generality of the results.

We use GT-ITM[23] to generate random network topologies for the simulations. We generate 5 native network topologies of 20 nodes each[2], 5 overlay topologies of 5 nodes each and 5 overlay topologies of 8 nodes each. This give us 50 possible combinations of mapping the overlay topology to the native topology at random. All observations in this chapter have been verified over these 50 synthetic topologies. However, we present the results from only one topology to monitor the trend accurately. The results observed for this topology are representative of those observed in the other topologies, unless otherwise mentioned.

The overlay links used in the simulation are bi-directional and we deploy a *ping*-like delay estimation scheme to determine the latency across an overlay link, i.e. we compute the round-trip time across each overlay link and halve the result to determine the one-way latency. This causes symmetric routing at the overlay layer, though the native TE-based routing over the set of directed links E is asymmetric.

We posit there are a few overlay nodes in a domain that exchange a certain amount of overlay traffic among each other. In addition, all nodes in the domain exchange a certain other amount of background traffic, i.e. background relative to the overlay network. The overlay traffic and the background traffic together represent the total load on the native

[2]We are restricted to a small native network owing to the huge computational complexity of the linear programming solution of MPLS-TE

network.

We vary the amount of traffic in the network by tuning two parameters – average link utilization u and the overlay traffic scale factor p. The former parameter determines the total load in the network and the latter determines the fraction of traffic on an overlay link that belongs to the overlay network. We configure the capacity of each native link in our topology to be 10Mbps. Once we determine the total load l (sum of all demands), we determine the source-destination pairs in the native network that are also part of the overlay. In all those pairs, we set the fraction of the overlay traffic to be p times the total load in those pairs. The remaining load becomes part of the background traffic. Once we know the total load in different sections of the native and overlay network, we randomly generate traffic demands between each source-destination pair. This traffic assignment is similar to that in [98][3]. We keep this traffic matrix the same all through our experimentation in order to analyze the dynamics for each load distribution.

It is worth noting that the overlay traffic matrix has to be combined with the overlay routes to determine the overlay networks' contribution to the real traffic matrix. An example of this operation can be seen in Fig. 28.

We configure the probing frequency in such a manner that each round of the interaction has one instance of TE, followed by three instances of overlay routing. After each TE, we update the latency of each overlay link and after each overlay routing operation, we update the real traffic matrix to reflect the changes. Thus, each layer's action influences the other layer's reaction.

In our simulation, we determine the exact routes for MPLS-TE by solving the linear program (LP) formulated in [55]. To solve this LP, we use the GNU linear programming kit[65].

We understand that our simulation setup can be considered simplistic i.e., it considers only a fixed native topology size of 20 nodes, intra-domain scenario and a single overlay network. However, we believe that this approach is well-suited to study each interacting

[3]There is a minor difference that we preset the total load first and then subtract the overlay traffic from it to determine the background traffic.

element. We reserve consideration of multi-AS multi-overlay case for future study.

5.3 Multi-layer Interaction: A simulation study

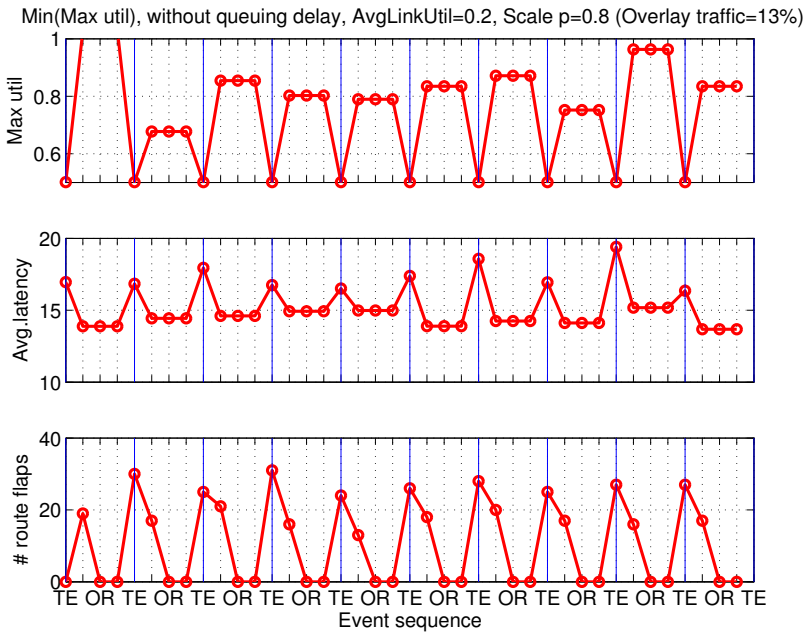
In this section, we present results from our simulation study of the interaction between overlay routing and traffic engineering, and discuss about the ideal routing choices for the leader.

5.3.1 Simulation Results

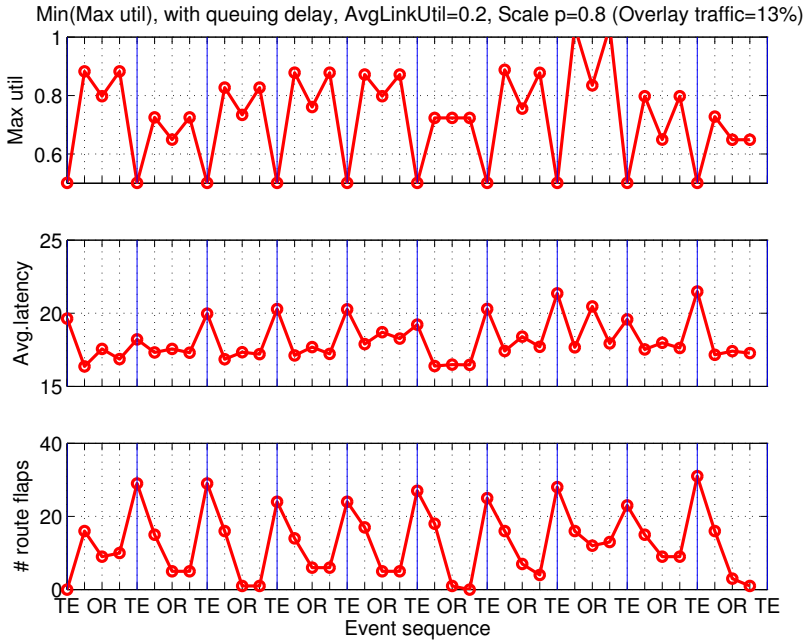
Fig. 29 presents results for the interaction between the two layers for a specific scenario, without applying any of our preemptive strategies. We observe that each TE procedure leads to an increase in the average end-to-end latency of existing overlay paths, while each overlay routing operation leads to an increase in the maximum utilization of the native network. This shows a clear conflict in objective between the two layers and gives sufficient reason for the instability. Owing to the probing frequency, we observe that the duration of sub-optimality for TE is longer in comparison to that for overlay routing. Though we plot only 10 rounds of interaction, the conflict in objective was noticed to extend even beyond 100 rounds.

The number of route flaps gives a numerical estimate of the instability in the network. We observe that the system suffers from persistent route flaps and does not attain a stable operating point even after 100 rounds elapse. This is accordance with earlier findings[100], which used both synthetic topologies and real tier-1 ISP topologies for verification. Note that the number of route flaps observed during TE serves as an estimate of the instability prevalent in all native routes, though the value we plot only represents the route flaps observed in the overlay traffic.

We observe that the amplitude of variation in the maximum utilization and average latency values is different between Fig. 29(a), where the queuing delay is negligible compared to the propagation delay, and Fig. 29(b), where queuing delay is comparable to the propagation delay. We used the M/M/1 formulation of queuing delay for lack of an accurate model. This is sufficient since we qualitatively seek a delay function that is inversely proportional to the available bandwidth on the link. Moreover, we are only focussed on



(a) Without queuing delay



(b) With queuing delay

Figure 29: The progression of observed performance for a particular topology with 8 overlay nodes and 20 native nodes. The above results represent the base performance, without using any of our strategies. Here, the objective of TE was to minimize the maximum utilization. Each TE event is followed by three overlay routing events.

comparing the temporal variation of the layer's outcome. On comparing Fig. 29(a) and Fig. 29(b), we observe that the introduction of queuing delay has reduced the amount of variation caused in the maximum utilization value. This can be attributed to the closed loop feedback inherent to queuing; when overlay routing picks a low delay link for multiple routes, it tends to increase the load on that link, leading to an increase in queuing delay, and consequently a cessation of using that link and a reduction of the load on that link. This indicates that the objectives of the two layers are less conflictive in the second scenario.

Another interesting feature in Fig. 29(b) is the flapping of overlay routes triggered by the change in queuing delay, which in turn was caused by the shifting of load during the previous overlay routing operation[4]. This shows minor unrest in overlay routing in the presence of substantial queuing delay, despite the absence of TE. In our simulation, the overlay layer did not employ any form of hysteresis to dampen these minor queuing delay-based route flaps.

We extended our analyses by varying different parameters of the simulation (namely, TE objective, number of overlay nodes, amount of overall traffic, percentage of overlay traffic, and the queuing delay), in an effort to completely profile the scenarios where the conflict is exacerbated. We observed a performance similar to those plotted above. The following is the summary:

- When the TE objective is to minimize the native cost, the trend is fairly similar to that shown in plot Fig. 29. The only difference is noticed when the queuing delay is non-negligible, wherein the objective of TE and overlay routing are lesser in conflict with each other. In that scenario, the objective of minimizing native cost tends to keep the load on all links low, thereby reducing queuing delay and consequentially the average latency of the overlay paths. On the other hand, overlay routing avoids overloading links, thereby reducing the native cost. Hence, the conflict is lesser, *yet significant*, in the case where the TE objective is to minimize native cost and the queuing delay is non-negligible.

[4]Note that any change in the overlay routing table leads to a change in the real traffic matrix seen by TE, though we do not change the native or overlay traffic matrix.

- We analyzed two sets of overlay topologies with 5 nodes and 8 nodes each. We observed that a higher fraction of overlay nodes cause higher conflict. This can be explained by inspecting the number of *multi-hop overlay paths*, defined as the overlay path which is not the same as the direct native route. A multi-hop overlay path is the primary reason why the native layer traffic matrix estimation is misdirected, i.e. when two nodes always communicate along the direct native route, then TE is able to load balance easily. Thus, *the higher the number of multi-hop overlay paths, the higher the conflict between TE and overlay routing.* In the 5-node topologies, we observed only 1 multi-hop overlay path. This caused the smaller topology to have no conflict, while the 8-node topology profiled above displays 20 multi-hop overlay paths, causing a higher level of conflict and a substantial number of route flaps.

- Irrespective of the size of the native topology and overlay topology, the occurrence of route flaps depends mainly on how conducive the overall network is to forming multi-hop overlay paths. Thus, we are justified in adopting a small native network, as long as there are sufficient overlay nodes.

- Increasing total load in the network stresses traffic engineering further and causes it to pick routes that are far more widespread. This worsens the link latencies as seen by overlay routing, giving it more reason to pick multi-hop overlay paths, thereby causing a higher variation in TE outcome. Hence, even at an average link utilization of 0.1, we start seeing considerable amplitude in variation.

- We analyzed the effect of overlay traffic by setting the scale factor at 0.4, 0.8, 1.2 and 1.6. We observe higher variations in the TE objective as the scale factor is high. When the amount of overlay traffic is very small, overlay routing has minimal impact on TE outcome. This is inline with our intuition and with past research results[98]. Consequently, we find very low amplitude of variation when the scale is 0.4 (which represents overlay traffic that is 6.4% of the total traffic).

5.3.2 Time-scale of Traffic Engineering Efforts

The previous subsection presented results for the cross-layer interaction at a relatively small time-scale, viz. TE is performed after three instances of overlay routing. This might be considered too frequent from the perspective of the ISP and represents the worst case scenario. In the event, TE is performed at a much lower frequency, then the native layer suffers a much longer duration of sub-optimality. The peak in the maximum utilization objective of Fig. 29 will persist for a long duration, although the route flaps may not persist. Despite the improve stability, this situation is still objectionable to the native ASes. Thus, irrespective of the time-scale at which TE operates, the native layer TE objective is disrupted by overlay routing and there is a need to adopt mitigation strategies, like those presented in Section 5.5.

5.3.3 Social Optimum

The social optimum is defined as the action-reaction pair that produces the best outcome for both the layers. This is a parameter that helps derive an estimate of the degree of non-cooperation (anarchy) in the network[88]. Ideally, the social optimum would be the desired operating point for both layers. However, lack of sufficient knowledge to exactly predict the other layer's response, makes it non-trivial to derive the social optimum. For instance, an overlay network that spans only a fraction of the native network, can only choose among the set of native routes it is exposed to and is unaware of a potential shorter route with lower load. Furthermore, the social optimum can also be inexistent in certain scenarios of conflicting objectives. Hence, we proceed to determine the best possible performance for a particular layer, even at the expense of the other layer.

5.3.4 Ideal Solution

Given infinite learning time and memory, one can create a map (as in Table 10) of peremptory actions for the leader, which cause a specific reaction by the follower and an associated change in outcome for the leader. Each entry can be created based on history and by trying out all possible routing decisions. The general idea is for the leader to replay any action

Table 10: Strategy profile for each input load/latency profile P_1

Initial	Leader	Effect	Follower	Outcome
P_1	Action A_1	Latency_1 Load_1	Reaction R_1	$\text{Latency}'_1$ Load'_1
P_1	Action A_2	Latency_2 Load_2	Reaction R_2	$\text{Latency}'_2$ Load'_2
P_1	Action A_3	Latency_3 Load_3	Reaction R_3	$\text{Latency}'_3$ Load'_3

that is known to yield a favorable (or favorable enough) outcome. It is worth noting that any choice in the strategy profile will be acceptable to both layers, i.e. when the leader picks a certain action, the final outcome after the reaction is bound to be agreeable to the follower. Clearly, there are intractable number of possibilities, for each load/latency profile. This makes it infeasible to determine in polynomial time the appropriate routing decisions for each layer.

In the following two sections, we propose preemptive strategies targeting a particular layer, with the assumption that the other layer does not deviate from its general objective. In addition, we assume that each layer has a general notion of the other layer's objective. Our strategies apply certain heuristics, based on the above study, to converge at a near-optimal routing table within polynomial time. Moreover, they do not require any other information besides what is report by basic network monitoring. Lastly, they require no cooperation or interface between the two layers, and exercise sufficient control over the follower indirectly. Thus, these strategies can be easily implemented in a realistic environment. All simulations results presented use native networks of 20 nodes, overlay networks of 8 nodes, average link utilization of 0.2, and an overlay traffic scale p of 0.8. However, the results were verified over all synthetic topologies generated, as explained in Section 5.2.

5.4 Overlay Layer Strategies

In this section, we present strategies that help the overlay layer preemptively steer the multi-layer interaction towards a converging point, wherein the performance of the overlay is almost as good as the case when there is no reprisal from the native layer TE. By making certain calculated routing decisions at the overlay layer, we ensure that TE does not get triggered; because its network monitoring has not sensed any change, or because TE does

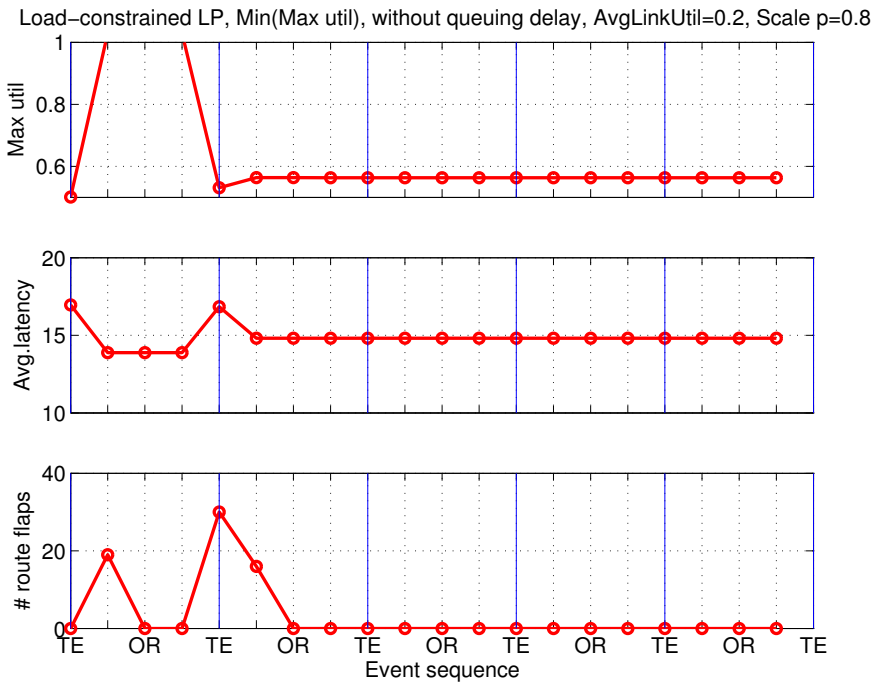


Figure 30: Performance results for the load-constrained LP strategy.

not find any alternatives besides the current routing table.

The strategies are classified, based on their nature towards the native layer, as *friendly* or *hostile*. The friendly strategy picks routes in such a manner that the TE objective is not altered much and the native layer still has a well-balanced load, while the hostile strategy performs extra operations to achieve the overlay layer objective by defeating the TE objective. The observations made in this section apply to both TE objectives and to both levels of queuing delay.

5.4.1 Friendly: Load-constrained LP

In this strategy, we make use of the fundamental idea that the TE sees only the real traffic matrix and not the end-to-end overlay traffic matrix. Hence, if we determine the load distribution in the network after TE's load balancing operation, and ensure that any future routing at the overlay layer always contributes the same load to the real traffic matrix, then TE has no reason to be triggered. Using this reasoning, we adopt the following algorithm for overlay routing:

1. Determine the available bandwidth on each overlay link using tools like Pathload[76]. Let us refer to the minimum of this value over all links as *min(availbw)*. We insure that the available bandwidth on all links are kept above this value.

2. Set the maximum allowable load on each overlay link to the amount of overlay traffic on that link, computed by a product of the overlay demand matrix and the overlay routing table. This is conservative because there might be more leeway.

3. Set the maximum allowable load on each unused overlay link a, i.e. an overlay link that transports no overlay traffic, to a value of *availbw(a) - min(availbw)*. This ensures that the TE objective is still respected.

4. Let $F_{(x,y)}^{(s,t)}$ represent the fraction of traffic between nodes s and t that goes over overlay link (x, y), and $\mathcal{L}(x, y)$ represent the maximum allowable load in the current round. Run the following linear program (LP), with the last additional constraint, to determine the overlay routes. This LP minimizes the sum of latency of each overlay path, while insuring that the load on each overlay link is within the allowable limit.

$$\textbf{minimize } \text{Total Latency} = \sum_{(s,t)\in V'\times V'} \text{latency}(s,t)$$

subject to:

$$\sum_{(x,y)\in E'} F_{(x,y)}^{(s,t)} - \sum_{(y,z)\in E'} F_{(y,z)}^{(s,t)} = \begin{cases} -1, & \text{if y=s} \\ 1, & \text{if y=t} \\ 0, & \text{otherwise} \end{cases} \qquad \forall y, s, t \in V'$$

$$\text{latency}(s,t) = \sum_{(x,y)\in E'} \text{delay}(x,y) \times F_{(x,y)}^{(s,t)} \qquad \forall (s,t) \in V' \times V'$$

$$\sum_{(s,t)\in V'\times V'} \text{overlay\_demand}(s,t) \times F_{(x,y)}^{(s,t)} \leq \mathcal{L}(x,y) \quad \forall (x,y) \in E'$$

Fig. 30 presents the simulation results for this strategy. In the first round of the simulation, we run a normal version of overlay routing (by setting all values in $\mathcal{L}$ to be ∞)

to get an estimate of the optimal overlay routing performance. After the first round, we run the LP-version of overlay routing with finite load constraints. We notice that the overlay is able to reduce the average latency achieved without causing an increase in the maximum utilization. Hence, it is a friendly strategy. Moreover, the above algorithm stabilizes within one round and requires data from only the previous round. The only drawback with this strategy being the added complexity in maintaining multipath overlay connections. The exact details of that are outside the scope of this thesis.

We also experimented with a *gradient projection* strategy, that shifts overlay paths from highly used overlay links to less used overlay links, while accepting a minor loss in performance. This serves as a form of load-balancing at the overlay layer, so as to prevent the TE from reacting. The reasoning behind such a strategy is that the popularity (betweenness) of certain nodes or links in an overlay network is much more than few others[137]. Hence, the *gradient projection* strategy tries to reduce this non-uniformity without causing a substantial increase in end-to-end latency. However, the simulation results indicated that the overlay routing performance achieved by deploying this strategy is inferior to that achieved by the *load-constrained LP* strategy.

5.4.2 Hostile: Dummy traffic injection

In this strategy, the overlay layer sends high amounts of dummy traffic on unused overlay links with the following two motives:

- Render TE ineffective: By sending high amounts of traffic, the overlay layer ensures that the objective of TE is stretched upto an extent where it becomes ineffective and has no effect on existing overlay routes. This gives the overlay layer complete freedom in picking routes and overloading certain links.

- Shift concern of TE: By sending dummy traffic, the overlay layer shifts concern of TE to the over-utilized native links and allows the other less loaded native routes to use the least possible resources. Thereby, making it more probable to obtain shorter native routes for the overlay links.

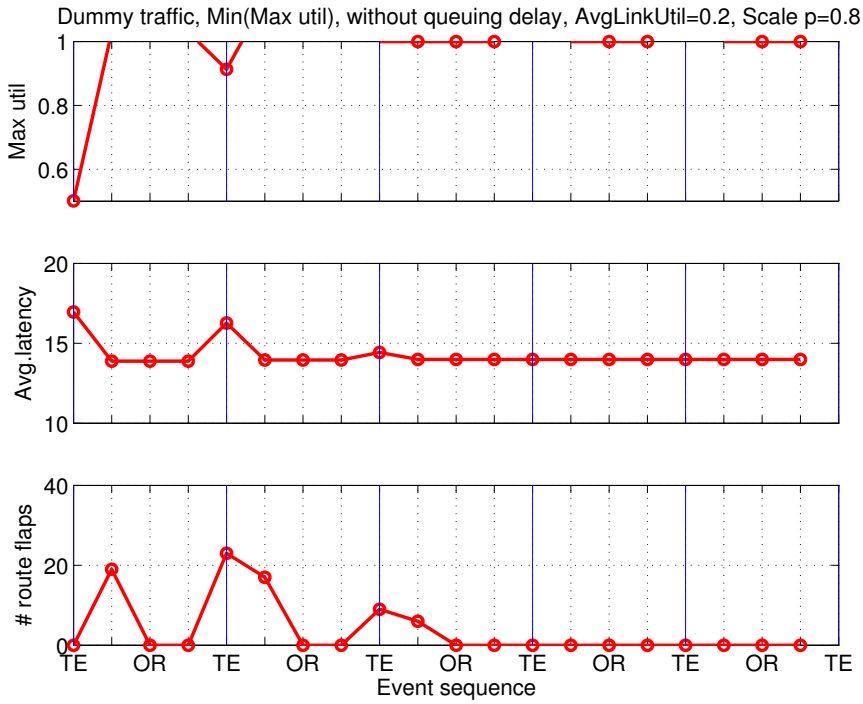


Figure 31: Performance results for the dummy traffic injection strategy.

Thus, sending dummy traffic prevents deterioration in overlay routing performance during future rounds. This strategy is counter-productive with regards to the overall system health. However, as long as the overlay links with dummy traffic do not intersect with overlay links we are truly concerned about, the risk incurred (in the form of queuing delay or packet loss) is minimal. Thus, we take special care that the links over which we send dummy traffic are i) unused by any overlay route, ii) non-overlapping with links under use. The latter constraint might require us to execute the *traceroute* program between the endpoints of each overlay link.

Fig. 31 plots the simulation results for the case where we choke unused non-overlapping links. We observe that the TE objective is completely violated, while achieving good performance for the overlay layer. The strategy was able to achieve good performance in the second round itself with no knowledge of previous load distribution.

We realize that this strategy may not fare as well in the case where multiple overlay networks, coexisting over the same native network, inject dummy traffic simultaneously. This is especially problematic when the dummy traffic sent by one overlay network enters

Table 11: Summary of inflation factor incurred by overlay strategies

Strategy	Overlay	TE
Friendly: Load-constrained LP	1.082	1.122
Hostile: Dummy traffic injection	1.023	1.992

links used for regular traffic of another overlay network. In such cases, it is not feasible to guarantee good performance for every overlay network's objective. We reserve the study of the multiple overlay network scenario for future work.

An artifact of using a *ping*-like protocol for latency estimation in our simulation is that the queuing delay of the reverse links also matter. Hence, we take special care that the overlay links over which we send dummy traffic are non-overlapping in both the forward and reverse direction.

5.4.3 Performance Comparison

Table 11 presents the values of the inflation factor incurred by each layer at the steady state, when the overlay layer is the leader. We observe that the two strategies attain close to optimal average latency values. Moreover, the stability of the overall system is greatly improved. From results in Fig. 30 and Fig. 31, we see that the native and overlay routes have no route flaps beyond round 2, indicating that the system attains a steady state within a few rounds.

The friendly strategy sacrificed some of its performance to prevent distortion of the TE's objective, while the hostile strategy achieved the best possible performance for the leader at the expense of the follower's performance.

5.5 Native Layer Strategies

In this section, we present strategies that help the native layer preemptively steer the multi-layer interaction towards a favorable converging point. Similar to the previous section, the strategies are classified, based on their nature towards the follower, as *friendly* or *hostile*. The observations made in this section apply to both objectives of TE and to both levels of queuing delay.

5.5.1 Friendly: Hopcount-constrained LP

In this strategy, we adjust the MPLS-TE formulation in such a manner that during each load balancing effort, it takes special care to keep the native routes at the same length as before. Thus, we insure that the overlay layer does not notice any change in the perceived overlay link latencies. This simple constraint on the native route length keeps overlay routing from being triggered, and helps retain the good load balance. To achieve this, we adopt the following algorithm at the native layer:

1. Let $f_{(x,y)}^{(s,t)}$ represent the fraction of traffic between nodes s and t that goes over native link (x, y). This fraction is the output of the MPLS-TE's LP formulation.

2. After each TE operation, compute the total hopcount $\mathcal{H}(s,t)$ of each native route (s,t). This can be computed as: $\sum_{(x,y)\in E} f_{(x,y)}^{(s,t)}$. This hopcount profile $\mathcal{H}$ tends to approximate the latency profile of the overlay layer.

3. Using the hopcount profile of the previous round as input, compute the new set of native routes that are of almost the same length. The LP of MPLS-TE, with an objective of minimizing the maximum utilization, can be augmented to enforce this constraint in the following manner:

$$\textbf{minimize } \text{Maximum util} = \max_{(x,y)\in E} \frac{\text{load}(x,y)}{\text{capacity}(x,y)}$$

subject to:

$$\sum_{(x,y)\in E} f_{(x,y)}^{(s,t)} - \sum_{(y,z)\in E} f_{(y,z)}^{(s,t)} = \begin{cases} -1, & \text{if y=s} \\ 1, & \text{if y=t} \\ 0, & \text{otherwise} \end{cases} \qquad \forall y, s, t \in V$$

$$\text{load}(x,y) = \sum_{(s,t)\in V\times V} \text{demand}(s,t) \times f_{(x,y)}^{(s,t)} \qquad \forall (x,y) \in E$$

$$\sum_{(x,y)\in E} f_{(x,y)}^{(s,t)} \leq 1.02 \times \mathcal{H}(s,t)_{\text{prev}} \qquad \forall (s,t) \in V \times V$$

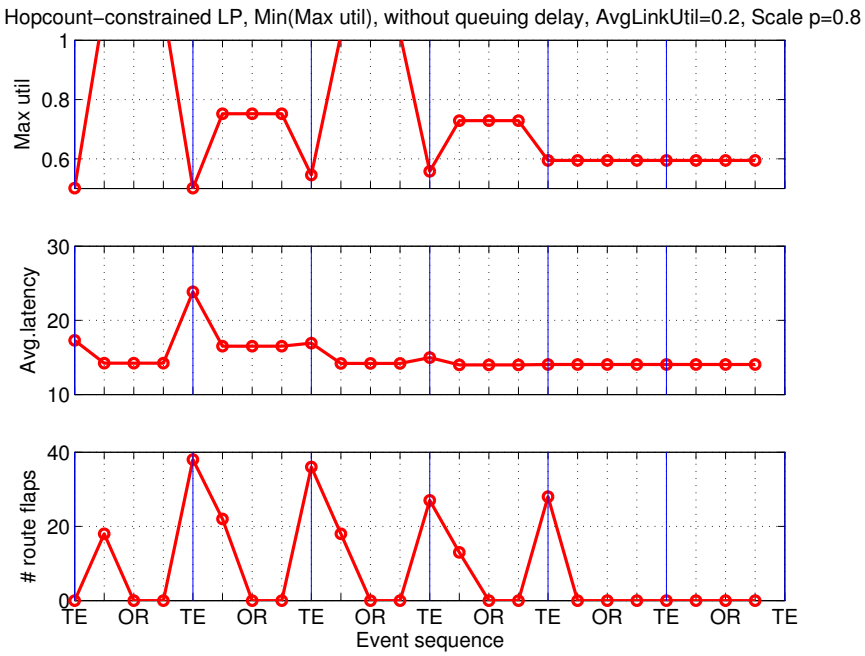


Figure 32: Performance results for the hopcount-constrained LP strategy.

The last constraint has been introduced to remove the need for overlay route change. We multiply the upper bound of the hopcount by 1.02 to allow the native layer a bit more flexibility in adjusting its routes, thereby allowing us to steer closer to the optimal load balance. Though, we restrict the hopcount and not the actual latency value, we reason that this approximation reduces implementation complexity and is sufficient to achieve good performance.

Fig. 32 shows the simulations results for this strategy. TE learns the hopcounts used by each native route over the first 4 rounds and eventually obtains a hopcount profile that correlates well with the overlay link latencies. At that point, the LP was able to balance the load and keep the overlay link latencies the same, thereby leading to steady state. Moreover, we were able to achieve good performance for both the layers, as observed in the plot of the two objectives.

We also experimented with an alternate strategy that tweaks the objective of traffic engineering, with an intent to attain stability indirectly. The goal of the strategy is to skew the load balancing process in such a way that it causes many links to achieve a link utilization in the range 0.3 - 0.6, and verify if this benefits system stability in the multi-layer

scenario. The fundamental idea is similar in spirit to the work in [135], that makes the case for *load unbalancing* in server task assignment to improve the fairness in scheduling. For this counter-intuitive *load unbalancing* strategy, we observed near-optimal TE performance, while causing the oscillations to converge in 7 rounds. However, its performance is inferior to the *hopcount-constrained LP* strategy.

5.5.2 Hostile: Load-based latency tuning

Traffic engineering is capable of adjusting the network to an optimal load distribution for most traffic matrices. However, it does not account for future alterations of the traffic matrix that may be committed by overlay routing. Thus, the ideal strategy towards retaining a good load balance is to restrict changes caused by overlay routing. This may be achieved by:

- Restricting the relay of overlay traffic in certain parts of the network in an effort to prevent the overlay layer from changing its current routes.
- Distributing load in such a manner that the overlay finds insufficient resources (or) high queuing delay on heavily used overlay links.
- Manipulating the latency (or any other metric that is of interest to the overlay) of all traffic on certain native links in such a manner that the overlay layer is offered an incentive or discentive to maintain the same routing table.

All these strategies lead to a deterioration in overlay performance and can potentially affect the experience of the end user. However, they have a difference in their motivation. The first two approaches discriminate against overlay traffic (and thereby raising concerns of net neutrality), while the third approach equally affects all traffic.

We implement the third approach in the following manner:

1. Constantly monitor the utilization on all native links.
2. If utilization is greater than or equal to 1, then increase the latency on the specific

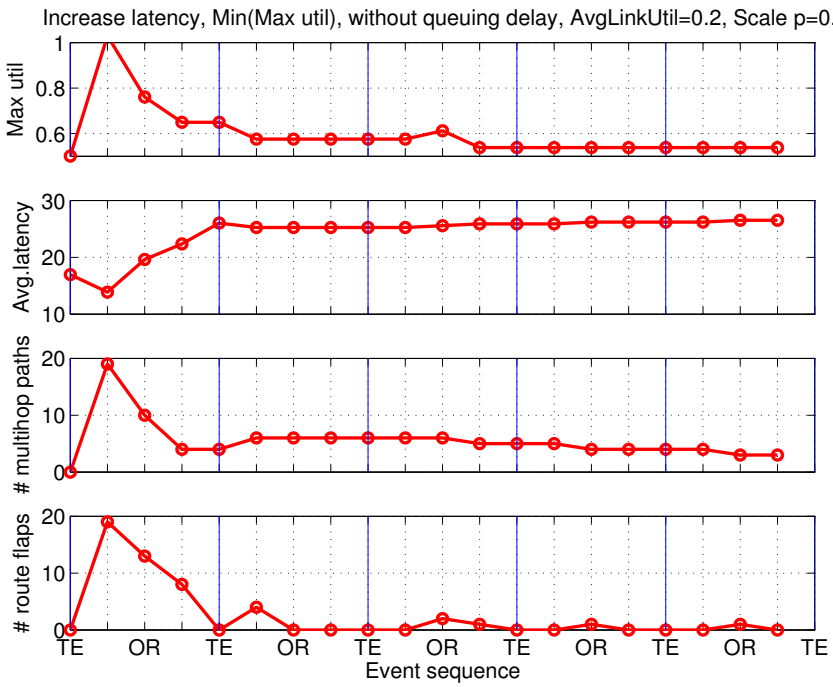


Figure 33: Performance results for the load-based latency tuning strategy.

link by 3 times a constant c ms[5].

3. Else if utilization is greater than the maximum utilization observed at the end of TE, then increase the latency on the specific link by the constant c ms.

4. Repeat this process until we attain an acceptable maximum utilization value.

The above procedure gradually learns which native links are key to overlay network and tends to dissuade usage of those links. When we simulated this strategy in our topology, we observed that the native layer was able to rapidly decrement the number of multi-hop overlay paths and attain steady state within 1 round. By indirectly increasing the latency, it avoided having to explicitly identify the overlay traffic in its network.

Fig. 33 shows the simulations results for this strategy when $c = 1$. We see that the native layer is able to achieve the best load balance at the expense of the overlay layer, and also to rid the system of further route flaps. Seemingly, this strategy is not as counter-productive as the hostile strategy proposed for the overlay layer in Section 5.4.

[5]We assume that the native layer is capable of increasing the latency of certain links by some means.

Table 12: Summary of inflation factor incurred by native strategies

Strategy	Overlay	TE
Friendly: Hopcount-constrained LP	1.027	1.184
Hostile: Load-based Latency tuning	1.938	1.072

5.5.3 Performance Comparison

Table 12 presents the values of the inflation factor incurred by each layer at the steady state, when the native layer is the leader. Both strategies yield a low inflation factor, indicating near-optimal performance for the leader. Similar to the results in Table 11, we observe that the hostile strategy achieves the best possible performance for the leader, viz. native layer, at the expense of the follower's performance. Moreover, the stability of the overall system is greatly improved when we use these preemptive strategies.

5.6 On Deploying Strategies at Both Layers

It is conceivable that the preemptive strategies can be simultaneously deployed by both the native and overlay layers. We present, in this section, the simulation results for the different combinations of strategies each layer can adopt. Specifically, we present results for the same scenarios analyzed in Sections 5.4 and 5.5: The objective of the native layer is to minimize the maximum utilization, there is no queuing delay, the native network has 20 nodes, the overlay network has 8 nodes, average link utilization is configured at 0.2, and the overlay traffic scale p at 0.8. However, the observations made in this section apply to the other scenarios as well.

Although each layer adopts a preemptive strategy, the interaction still remains a sequential repeated game. Each layer performs its own network monitoring operation and reacts to any changes it detects, in order to achieve its local objective. However, we assume that the decision process of choosing the strategy remains independent i.e., *Each layer only has a general knowledge of the other layer's objective and not the other layer's strategy.*

5.6.1 Friendly Overlay and Friendly Native

We first analyzed the case when the overlay layer adopted the *load-constrained LP* strategy and the native layer adopted the *hopcount-constrained LP* strategy. Fig. 34 shows the

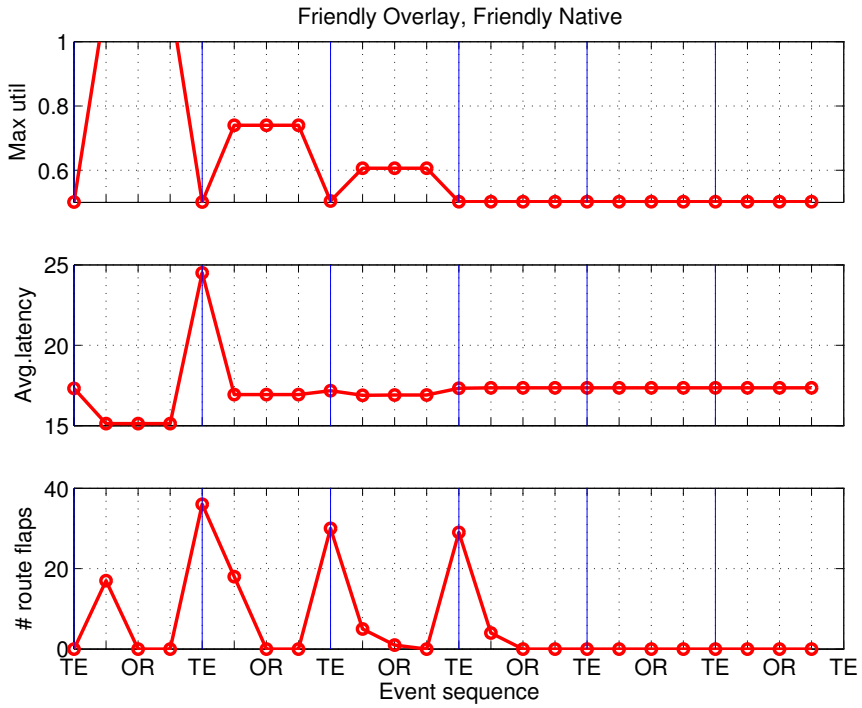


Figure 34: Performance results when both native and overlay layers adopt a friendly strategy.

simulations results for this combination. We observe that the performance achieved by TE is the best it can be, while that achieved by overlay routing is sub-optimal relative to when native layer did not adopt any strategy. This can be attributed to the fact that the native layer tries to approximate the latency profile by a hopcount profile. Thus, after both layer deploy preemptive strategies, the system stabilizes at a point where each overlay link is slightly longer than its earlier value, causing the higher inflation.

5.6.2 Hostile Overlay and Friendly Native

We next analyzed the case when the overlay layer adopted the *dummy traffic injection* strategy and the native layer adopted the *hopcount-constrained LP* strategy. Fig. 35 shows the simulations results for this combination. We observe that the overlay layer performs well, while TE suffers from the hostile action of overlay layer. This is expected behavior as the overlay layer disrupts the primary objective of the native layer, despite being guaranteed short routes by the friendly strategy of the native layer.

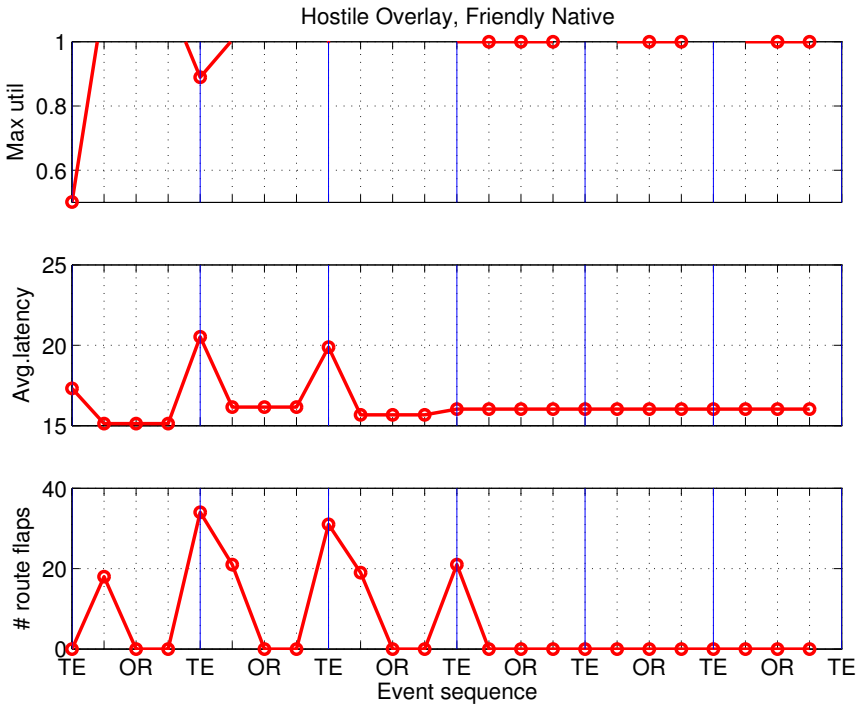


Figure 35: Performance results when the overlay layer adopts a hostile strategy and the native layer adopts a friendly strategy.

5.6.3 Friendly Overlay and Hostile Native

We next analyzed the case when the overlay layer adopted the *load-constrained LP* strategy and the native layer adopted the *load-based latency tuning* strategy. Fig. 36 shows the simulations results for this combination. We observe that the overlay layer is sub-optimal relative to when native layer did not adopt any strategy, while TE achieves good performance within a few rounds However, the inflation is not as high as it was in Section 5.5.2. This is because the *load-constrained LP strategy* adapts itself to the current environment and does not exert as much load on the native layer. Thus, the *load-based latency tuning* has minimal effect on the latency of overlay links, causing a low inflation of the overall latency.

5.6.4 Hostile Overlay and Hostile Native

Lastly, we analyzed the case when the overlay layer adopted the *dummy traffic injection* strategy and the native layer adopted the *load-based latency tuning* strategy. Fig. 37 shows

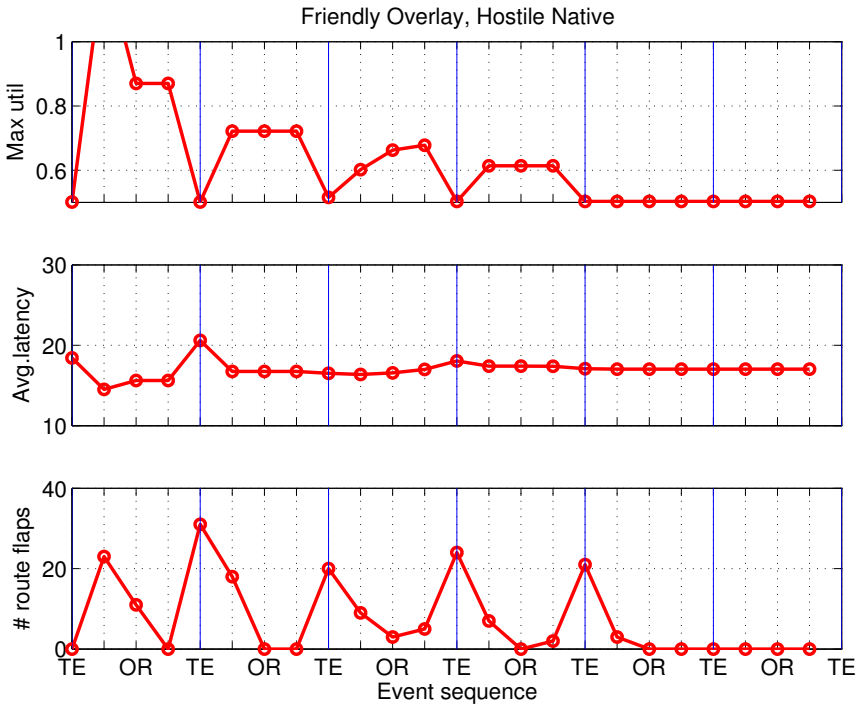


Figure 36: Performance results when the overlay layer adopts a friendly strategy and the native layer adopts a hostile strategy.

the simulations results for this combination. We observe that each layer suffers from the hostile action of the other layer. Though the system reaches a stable operating point within a few rounds, the objective of both layers are disrupted.

5.6.5 Performance Comparison

Table 13 presents the values of the inflation factor incurred by each layer at the steady state, when both layers adopt a preemptive strategy. We observe that the performance achieved by each layer mainly depends on the other layer's strategy; The simple rule of thumb being that a layer's performance deteriorates when the other layer adopts a hostile strategy. Nevertheless, the stability of the overall system is greatly improved when we use these preemptive strategies.

The formulation in Table 13 is similar to the payoff seen with the *Prisoner's dilemma* game, a non-cooperative repeated game where the prisoners need to cooperate to achieve the best possible payoff and the player that cooperates without reciprocation incurs poor payoff relative to the other player[57]. This motivates the need for cooperation between the

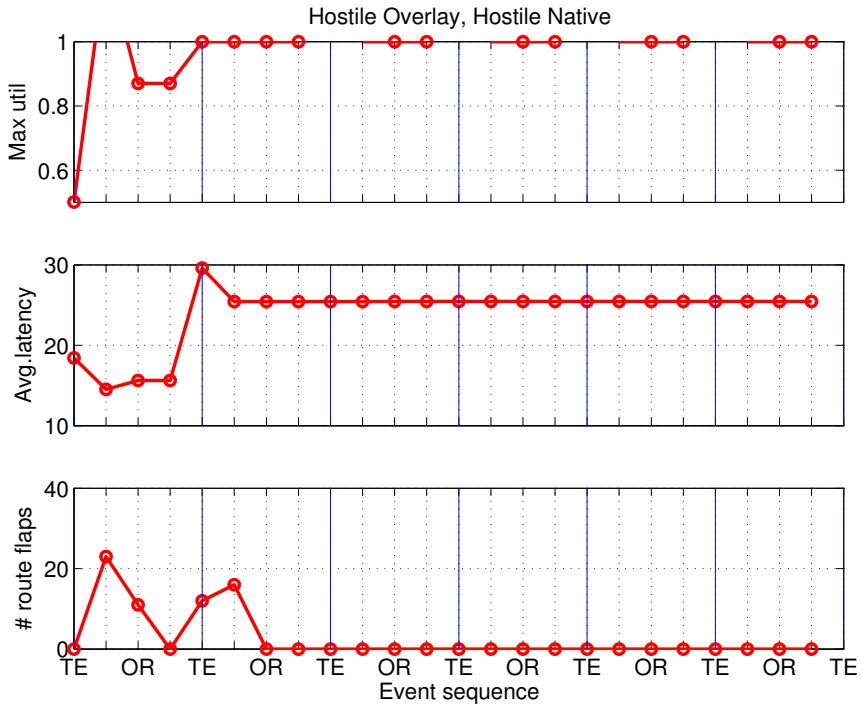


Figure 37: Performance results when both native and overlay layers adopt a hostile strategy.

Table 13: Summary of inflation factor incurred by deploying strategies at both layers

Strategy		Performance	
Overlay	**Native**	**Overlay**	**Native**
Friendly	Friendly	1.269	1.002
Hostile	Friendly	1.172	1.995
Friendly	Hostile	1.344	1.003
Hostile	Hostile	1.863	1.995

two layers.

5.7 Related Work

Our work builds directly on [100], which addresses the question of whether the multi-layer system with conflicting objectives will reach a steady state. However, unlike our work, they do not derive means to resolve the detrimental effects. Another related work analyzes the multi-layer interaction as a Nash routing game with each layer aiming to optimize its cost[98]. Even though the two players in the system are non-cooperative, their objective is similar. However, this similarity in objective is unrealistic in most scenarios where no direct transformation exists between the application specific metric (like end-to-end latency) and

native link load.

Yong et al.[176] propose proactive overlay routing algorithms that provide shortest paths, while constantly improving the headroom on each link. This tends to have the same effect as minimizing the maximum utilization, thereby reducing the conflict in objective. However, this solution is plagued with instability issues, which is precisely what our strategies eliminate.

Korilis et al.[87] investigate using Stackelberg approaches for optimizing the overall network performance, by deploying a manager that distributes its traffic after predicting the response of the other users in the network. However, in their work, the objective of all players are aligned and reflect the M/M/1 delay function. Hence, it is not as applicable when the objective is conflicting. Moreover, they assume knowledge of the follower's response (input + objective function), which is not feasible in reality. Similarly, the work in [173], which uses a Stackelberg approach to improve the performance of the overlay layer, also assumes complete information of the follower and a M/M/1 cost function for the two layers. Our preemptive strategies do not make these assumptions, and considers conflicting objectives between the two layers, without requiring complete information about the other layer. Furthermore, they use a gradient projection search to obtain an approximate solution, which is locally optimal and closer to the initial point, in one iteration. In contrast, our strategies arrive directly at the optimal choice within a few rounds, and is unrestricted by the original solution space.

5.8 Summary

In this chapter, we investigate the multi-layer interaction between overlay routing and traffic engineering, and propose strategies to steer this interaction towards an improved routing performance for one of the two entities. The motivation behind our work is the sub-optimality and instability, caused by the two layers having a conflict in objective and repeatedly altering their routes to achieve selfish goals.

The strategies we propose make one layer a leader and the other layer a selfish follower. In such a scenario, it is possible for the leader to achieve its desired performance within a few

rounds of the interaction, thus ridding the network of the inherent instability. Specifically, we propose two strategies for the overlay layer: *load-constrained LP* and *dummy traffic injection*, and two strategies for the native layer: *hopcount-constrained LP* and *load-based latency tuning*. From simulation under various network conditions, we observe that our strategies achieve near-optimal performance and converge within a few rounds of interaction, for both the overlay and native layers. Our preemptive strategies i) are simple and easily deployable, ii) do not require any cooperation or interface between the two layers, and iii) work with negligible information about each layer.

Some of our strategies breach the follower's objective, leading to unavoidable sub-optimality in the follower. We call such strategies hostile. We observe that the leader achieves the best performance when deploying a hostile strategy, showing a higher level of selfishness. The hostile strategies are counterproductive, while the friendly strategies improve the performance of both layers. We acknowledge that the solution space of hostile and friendly strategies is generally vast and that our work primarly offers a preliminary proof-of-concept.

CHAPTER VI

CROSS-LAYER INTERACTION OF PERFORMANCE-AWARE OVERLAY APPLICATIONS

6.1 Introduction

BitTorrent[16] is a peer-to-peer (P2P) protocol that has achieved remarkable popularity as a well-performing file-sharing application. The popularity is at such a level that some estimate the BitTorrent traffic to constitute nearly 35% of the overall Internet traffic[21]. The main property that fueled this attractive performance is the improved tit-for-tat incentive mechanism that reduces free-riding and increases user cooperation. Using this mechanism, the service capacity of the BitTorrent network increases with the demand for the content, thereby making the system highly scalable[170]. Besides the popularity, BitTorrent applications are also notable for their aggressive performance-awareness, i.e. the peer selection is primarily regulated by measurements of the achieved download or upload rate.

There is a downside to the surge in the number of (or the volume of traffic generated by) these *performance-aware* overlays, which we define as virtual networks that adapt their functioning and routes based on measurements or estimates of the state of the underlying native network. This adaptation is a selfish approach that can have serious impact on the dynamics of the native layer. This is similar in spirit to the impact observed with service overlays in the past (like conflict with traffic engineering[100, 98, 139], conflict with inter-domain policy violations[137]). Fig. 38 explains the reason behind this similarity. We observe that the process of choosing the appropriate peer to download data from can be portrayed as a routing decision problem. For instance, host A_1 can choose to route the content through the intermediate hop of A_3 or A_4 based on the perceived performance, with each decision having different effect on the underlying native layer. Furthermore, there can be multiple "routes" using the same access link, thereby contending for resources with each other and with other legacy traffic. Such contention, in the presence of load-balancing

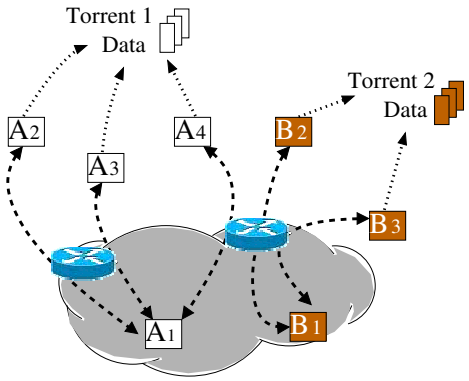


Figure 38: BitTorrent peer-selection is equivalent to overlay routing as each host determines the optimal route to the desired data.

efforts, can cause instability and cross-layer conflict.

In this chapter our goal is *to investigate the cross-layer interaction between BitTorrent file-sharing and native layer traffic management operations*, since BitTorrent traffic represents a large category of traffic transiting today's Internet, with tremendous potential for future growth. Specifically, we investigate the interaction with the following proposals for reducing the overall inter-AS traffic and the peak load on each inter-domain link:

- Inter-domain traffic engineering (TE) with an objective of minimizing the maximum utilization across access links[46]
- Throttling or rate limiting the aggregate BitTorrent traffic to regulate their impact[8]
- Manipulating BitTorrent peer selection to favor local peers without impacting the user experience[21, 134, 80, 50]

Although cross-layer interaction in BitTorrent systems has been investigated from the perspective of Internet Service Provider (ISP) cost [140, 80, 50], our work is the first to investigate the impact of BitTorrent applications on native layer traffic engineering. The interaction between BitTorrent file-sharing and intra-domain TE is minimal because the peers are often located outside the domain and the traffic variability caused by downloading from intra-domain peers is very low. Thus, in this chapter, we only focus on the interaction with inter-domain TE.

We use both real traces of BitTorrent activity observed in the Georgia Tech network and a comprehensive discrete-event simulation of BitTorrent activity in the Internet to obtain insights into: 1) the fluctuation of overlay traffic between inter-domain access links, 2) the macroscopic behavior of BitTorrent clients with regards to peer and piece selection, 3) the effect of multiple coexisting BitTorrent networks on the native layer. We show that the BitTorrent protocol easily disrupts the load-balancing performed by TE, while being manageable by locality-based traffic management and bandwidth throttling. Although the latter two native layer strategies perform well in reducing the impact of BitTorrent traffic, they cause a deterioration in user experience and incur additional infrastructure costs. This makes the three native layer strategies unattractive for deployment.

To resolve this dilemma, we *propose and investigate* layer-aware *BitTorrent strategies (that use awareness of the dynamics at each layer) to mitigate the detrimental effects caused by the cross-layer interaction.* Specifically, we propose the following two approaches, each of which uses different levels of information about the native layer:

- Tuning the BitTorrent protocol to reduce its aggressive behavior without any information of the native layer
- Performing layer-aware decisions in peer and piece selection based on knowledge of peer location

Our simulation studies indicate that the BitTorrent strategies we propose are simple, inexpensive and effective in reducing the peak load on the inter-domain links without compromising on BitTorrent performance.

It has been established by numerous studies that the usage of BitTorrent traffic is further increasing, partly due to its adoption by legitimate content providers for reducing their infrastructure costs[74, 148]. Considering both the current and impending volume of BitTorrent traffic, it is in the best interest of ISPs to rethink their traffic management model. Furthermore, adopting a layer-aware approach at the BitTorrent layer is crucial to eliminate the conflict and achieve a stable operating point that is mutually agreeable.

The remainder of this chapter is organized as follows: We describe the BitTorrent protocol and inter-domain TE, along with the issues involved in the cross-layer interaction, in Section 6.2. We present the model of the two players and our simulation results of the base interaction in Section 6.3. Section 6.4 investigates the effect of native layer traffic management strategies on the performance of BitTorrent. We present our layer-aware approach to reduce the impact of BitTorrent applications in Section 6.5. We briefly describe related work in Section 6.6. We summarize the chapter in Section 6.7.

6.2 Multi-Layer Interaction

In this section, we provide a brief description of the BitTorrent protocol and the issues involved in its cross-layer interaction with the native layer. Further, we present observations from traces collected in our campus network that motivates the need for further investigation.

6.2.1 BitTorrent Protocol

BitTorrent[16] is a peer-assisted file-sharing protocol that employs a tit-for-tat (TFT) incentive mechanism to reduce free-riding and increase user cooperation. Unlike earlier P2P networks, the richness and efficiency of the overall BitTorrent system is less dependent on the peer churn (arrival, abort and departure). Instead, there are other performance-aware policies determining if we can obtain content from a certain peer. These policies have been shown, both analytically[125] and empirically[94] to boost the efficiency of the system significantly.

Each file distributed in the BitTorrent network uses a unique *torrent* to identify and describe the content. A file being shared in the BitTorrent network is divided into many *pieces* of about 256KB (more of an implementation consensus), with each piece further divided into 16 *sub-pieces* of 16KB each, thereby enabling more efficient pipelining of requests[31]. A peer that wishes to procure the content reads the corresponding torrent file and contacts the *tracker*, which is a centralized host tracking the membership of each torrent and regulating the neighborhood of individual peers.

Each peer in the system follows a dichotomous existence: as a *leecher* it is in the constant

process of attempting to obtain the complete file and as a *seed* it sources the complete file to other leechers. The duration each peer spends in either state often depends on the peer's volition, rather than on network characteristics[68]. When a leecher enters the system, it contacts the tracker and obtains a list of (maximum) 50 random peers that will act as its initial set of neighbors[1]. Then, it makes a persistent TCP connection to each of its neighbors, who add the leecher onto their neighbor set, and learns about the pieces they have. Each neighbor also sends in updates on the list of pieces they have, as and when they have a new complete piece.

Although a peer A may have more than 50 neighbors, it only uploads to 5 *interested* leechers among them; An interested leecher is a peer that wants some content that peer A has. Of these 5 leechers, 4 (default) are selected based on their attractive transfer rate (upload rate if peer A is a seed, or download rate if peer A is a leecher). To achieve this signaling, the peers use a *choke–unchoke* protocol, whereby it indicates if it is ready to accept requests from the other end. Besides the 4 leechers *unchoked* based on performance, 1 leecher is picked randomly from the other interested neighbors. We call this as the *optimistic unchoke*. This enables the peer to learn about other peers in the neighborhood, thereby improving the chance of picking the best performing peers. Once unchoked, a leecher attempts to obtain the piece that is rarest in its neighborhood. This way it can improve its chances of being favored by other peers. This *rarest piece first* policy is known to bring sufficient diversity into the system[94].

In order to be picked for downloading content, a leecher needs to have uploaded content at an attractive rate. This is how the TFT reciprocation mechanism works. However, in the case of seeds, we are only concerned about propagating the content as fast as possible and do not have a reciprocation strategy. Note that the transfer rate is computed as the average of traffic volume transferred over past 20 seconds. BitTorrent is quite aggressive in constantly re-evaluating the set of peers unchoked, the default periodicity of re-evaluation being 10 seconds. Further, the same peer may be unchoked by more than 4 other peers

[1]The tracker does not possess knowledge of pieces in individual peers and often does not concern itself about the location of the peers, thereby making the neighbor set generation completely random.

Table 14: Illustration of BitTorrent dynamics at a leecher. Each box lists ID and download rate (in Mbps) of peers unchoking it

	Link	$t = 60$ regular unchoke	$t = 61$ optimistic unchoke	$t = 100$ regular unchoke	Final load (Mbps)
When neighbors share bottleneck	I	A: 2 B: 2	A: 1 B: 1.5 C: 2	 B: 1.5 C: 2.5	4
	II	D: 2 E: 2	D: 2 E: 2	D: 2 E: 2	4
When neighbors do not share bottleneck	I	A: 2 B: 2	A: 2 B: 2 C: 2.5	A: 2 B: 2 C: 2.5	6.5
	II	D: 2 E: 2	D: 2 E: 2	D: 2	2

because the choking protocol and neighbor set of each peer is independent.

There are a few other minor strategies in place to achieve faster completion and prevent deadlocks, viz. random piece first, end-game mode, anti-snubbing. We describe them as and when needed. Refer to [31, 94, 48, 15] for a detailed description of the basic protocol and other proposed enhancements.

6.2.2 Cross-Layer Interaction

Each entity described above operates solely on the results of their own network monitoring process without any concern for the dynamics in the other layer. Typically, the objective of the native layer is to provide as much headroom as possible on each access link at an acceptable cost, so that there are ample resources for a broader range of higher layer users. This operation is referred to as *load-balancing*. In contrast, the objective of the BitTorrent protocol is to obtain as much bandwidth as possible for its own content transfer and to minimize the overall time taken to finish downloading content. This mismatch in objective between the two layers leads to contention for limited physical resources and cross-layer conflict. Thus, independent operation of several torrents at the higher layer can to lead disproportionate usage of different inter-domain links, thereby disrupting the load balance and resulting in high maximum utilization ($\frac{\text{load}}{\text{capacity}}$) on some links. These are the issues we primarily focus on in this chapter.

Table 14 illustrates this contention better[2]. Consider a leecher X unchoked by 4 peers A, B, D and E. During this unchoke period, let us say the leecher is optimistically unchoked by a peer C with higher upload rate. When the downloads are all bottlenecked at *Link I*, this causes the download rate of other peers sharing the link to deteriorate. Subsequently, X will reciprocate only to the best neighbors and start choking peer A. This causes peer A to perform TFT and choke leecher X, thereby causing the load on *Link I* to stay within earlier bounds. This indicates the presence of a feedback loop preventing over-utilization and causing constant swapping of peers. However, when the downloads are not bottlenecked at *Link I*, there is no change in download rate of peers A and B. This can possibly lead X to retain peers A and B, and choke peer E instead, thereby causing the load on *Link I* to be much higher than that of *Link II*. Thus, when the inter-domain links are not bottlenecked, BitTorrent performance-awareness causes unfair usage, and when the inter-domain links are bottlenecked, they aggressively consume all available bandwidth.

When the BitTorrent protocol makes rapid routing decisions in an adaptive manner, the other application traffic goes through undesired changes in network characteristics. Furthermore, the load variation introduced by the BitTorrent applications often instigates load-balancing efforts at the native layer to react, thereby causing route oscillations for other unrelated background traffic. Depending on the time scale of the load-balancing operations, this frequent route changes can potentially lead to instability in the system and a deterioration of end-to-end performance.

The following features of BitTorrent are the main reasons for the cross-layer conflict:

- Tit-for-tat mechanism: This induces the performance-awareness in the BitTorrent protocol and causes the host to exchange content with selective peers. This deterministic behavior can potentially over-utilize certain access links.

- Geographical diversity of peers and their flashcrowd arrival pattern: There exist many torrents that have a far reaching membership and a relatively high level of popularity, causing flashcrowd effects frequently[47]. Furthermore, the users of BitTorrent

[2]In this example, we do not consider the effect of TCP congestion control.

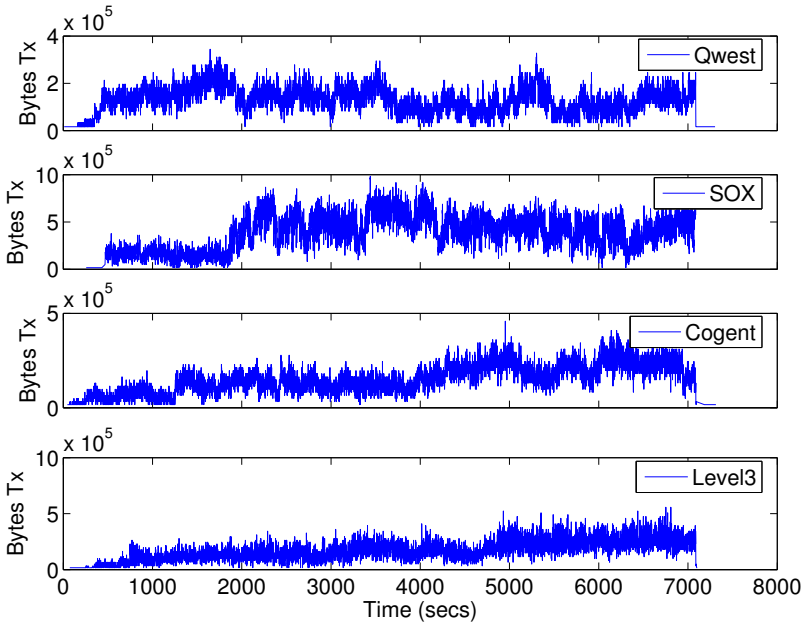


Figure 39: Traffic variation across the 4 upstream providers caused by our client.

applications often arrive in a flashcrowd pattern once the content is published.

- Seed uploading decisions: The only objective of the seed is to upload content to those peers capable of receiving content at a high rate. Thus, any seed within an AS ramps up its uploading rate to the maximum available bandwidth, thereby increasing the utilization of those access links drastically.

- Unfairness in contributions: BitTorrent has been shown to be altruistic in offering content without achieving a gain in downloading rate and to cause an unfairness in the amount of content served by each node [14, 48]. This can also translate to disproportionate usage across access links.

The random decisions like neighbor selection by the tracker and optimistic unchoking to explore the neighborhood do not cause load fluctuations because of its oblivious nature. However, they can be tuned to better cooperate with native layer dynamics. We discuss these options in Section 6.5.

6.2.3 Measurement-based Study

To get an idea of the non-cooperative interaction between the two layers, we monitored the download of 4480.8MB of the Debian stable release `4.0_r0` on Tuesday May 29, 2007. This is 52 days after the release and cannot be considered as a flash crowd scenario. The total download took about 7334 seconds, during the process of which the client inside Georgia Tech downloaded content from about 450 different hosts spanning 253 different ASes, none of them being Georgia Tech itself. Furthermore, the client contacted the tracker 11 times to obtain a new set of neighbors. However, the number of new peers it actually learns from the tracker declined gradually. As the neighborhood constituted both peers the Georgia Tech client contacted and those peers that contacted the client, the neighborhood remained fairly stable all through. We plot the usage level of the four different providers this BitTorrent client used over its lifetime in Fig. 39. We see the ingress usage level experienced substantial variation over the lifetime and unfairness across providers. We notice similar behavior with the egress usage as well. However, the egress usage displays less variation than the ingress usage because the number of simultaneous uploads at any point of time is restricted to 5 (including the optimistic unchoked peer). Georgia Tech being an academic network with good bandwidth subscriptions makes it easier for the client to pick the best downloader / uploader and keeps the set of unchoked peers relatively stable.

In Fig. 40, we plot the ON/OFF behavior of download from the top 10 uploaders. We notice that three peers are almost always uploading content to our client, while the others go through pause phases. As the Georgia Tech client is hosted in a network with good bandwidth in either direction, peers outside our network had a natural inclination to serve content persistently to our client. However, our client was not persistently interested in some other peers. This decision of peer and piece selection is the main cause of fluctuation in the utilization across different access links. Another reason for the fluctuation is the prolonged learning period a leecher spends in learning about its neighborhood. During this phase, it associates with many potential uploaders and is unconcerned about network-level repercussions.

Further, to get a glimpse of the overall BitTorrent activity seen by the border router, we

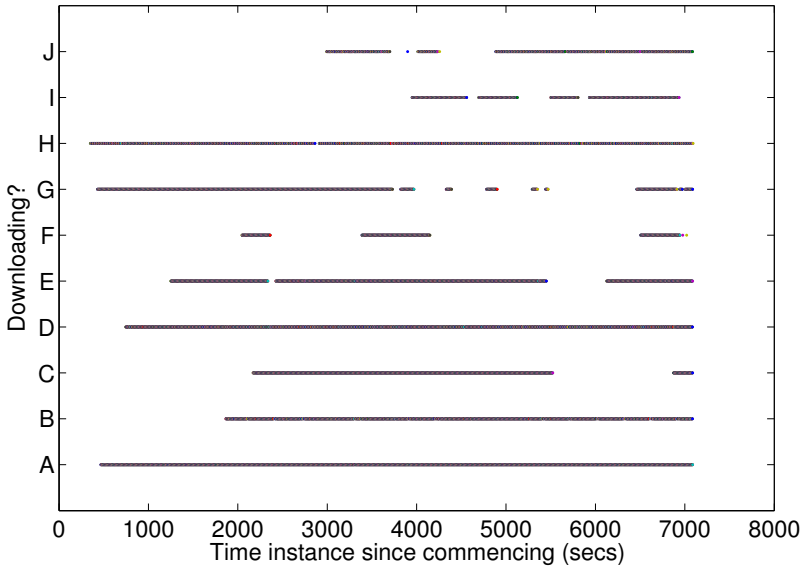


Figure 40: The ON/OFF behavior of top 10 uploaders to our Georgia Tech client. A point on the graph corresponding to a particular peer indicates that downloading happened at that time instance.

inspected regular traffic captured at the Georgia Tech ingress and egress router on Tuesday April 4, 2006[3]. We plot the ingress usage of overall BitTorrent activity for Georgia Tech in Fig. 41. We notice distinct periods when the BitTorrent activity peaks in different access links, indicating a need for load-balancing and the need for further investigation into the multi-layer interaction.

6.3 Evaluation of BitTorrent vs Native Layer Dynamics

In this section, we describe the performance metrics used for each layer and present our methodology of evaluating the base interaction between BitTorrent and the native layer. We adopt a simulation-based approach because the BitTorrent protocol is less predictable and highly complex to model accurately.

6.3.1 Performance metrics

The routing performance achieved by native layer traffic management at an AS can be measured by:

[3]Though this packet capture does not actually overlap with the previous experiment we believe the trend to be similar.

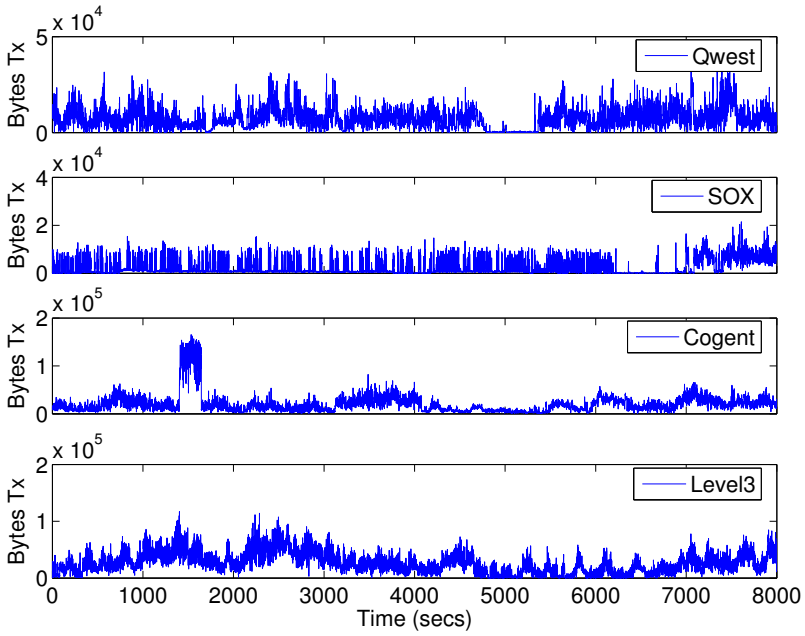


Figure 41: Traffic variation across the 4 upstream providers caused by the overall BitTorrent activity.

- *Maximum utilization* seen across all inter-domain links. It can be computed as $\max_{a \in E} \frac{X(a,\delta)}{C(a)}$, where a represents a link in the set of inter-domain links E of this AS, $X(a, \delta)$ represents the average ingress traffic rate (or egress traffic rate) over a time interval of δ, and $C(a)$ represents the capacity in that direction. As ISPs are often concerned about the average usage over different time intervals, the metric parameter is also designed to include this dependency. A smaller δ is more susceptible to variation in traffic rate and may report higher than desired utilization value. As mentioned earlier, this factor covers many of the metrics ASes might be potentially interested in.

We measure the performance of the BitTorrent system by:

- *Average finish time inflation*: This represents the ratio of increase in finish time incurred by each leecher in the AS because of adopting a traffic management strategy; We measure the finish time as the amount of time required to complete the download, without including the time the peer may later serve as a seed. Thus, the average finish time inflation is computed as $\frac{1}{N_l} \sum_{i=1}^{N_l} \frac{\Delta_i'}{\Delta_i}$, where N_l represents the number of leechers

in this AS, Δ_i and Δ'_i represents the finish time seen by leecher i before and after strategy respectively. This factor directly represents the aggregate user experience of hosts in the AS.

6.3.2 Simulation Setup

Simulating the interaction of the overall BitTorrent system, along with the native layer dynamics, is a complex task involving several components. We use a discrete-event simulator to evaluate the cross-layer interaction. The performance achieved by the BitTorrent application and the native layer depend on several parameters like the distribution of peers, data availability, P2P churn, and the load at different parts of the native network. Hence, we simulate multiple runs with different choices of ASes hosting BitTorrent peers (called *host ASes*) and a random distribution (both density and location) of peers over this native topology, along with varying levels of background traffic, to improve the generality of the results.

6.3.2.1 Modeling Native Layer Dynamics

In each run, we randomly picked 100 ASes, each with different levels of multi-homing, from a list of 21,416 ASes observed in a large set of actual BGP route dumps obtained from 6 RouteViews servers[131], 14 RIPE RCCs[128], 30 public routeservers and 1 lookingglass server[99] in November 2005. Of the 100 host ASes, we ensured that 60% of them belonged to non-Tier-1 provider ASes, in order to simulate the presence of home users. The maximum out-degree of the host ASes was bound to 6.

We computed the AS-level route of each overlay link by using the BGP routes collected from multiple vantage points as input and performing policy routing between the two host ASes i.e., we computed the shortest AS-path that does not violate native layer policy[4]. In order to obtain the inter-AS relationships that policy routing needs as input, we adopt Gao's algorithm[59], supplemented by the partial AS relationship information[166], and our own heuristics to eliminate most of the algorithm's inaccuracies[137].

[4]This is an approximation as the actual routing tables and the policies are unavailable.

As is often the case with BitTorrent data transfer, the bottleneck of each transfer is the access link of the host AS. The inter-domain link associated with each stub AS was assigned a symmetric capacity of 100 Mbps. Each home-user was assigned an asymmetric capacity of 1Mbps upload and 10Mbps download, and each host inside a stub AS was assigned a (intra-domain) capacity of 100 Mbps. This capacity assignment allows us to model the heterogeneity of peers in the BitTorrent network. We then randomly generated the level of background traffic on each access link. We keep this volume of background traffic the same all through the run so as to clearly identify BitTorrent dynamics as the main reason for load fluctuations. The maximum downloading rate between any pair of ASes will be the minimum of the available downloading bandwidth at one end and the uploading bandwidth at the other end.

The download / upload rate for each peer-to-peer connection of a particular torrent is computed as the fair-share of the available bandwidth, where the available bottleneck bandwidth is shared equally among flows not bottlenecked elsewhere (independent of the round trip times of each connection). We are justified in assuming fair-share allocation because 1) BitTorrent downloading represents a persistent bulk transfer which will be able to saturate the available bandwidth[76], and 2) many implementations of TCP have mechanisms to ramp up the sending rate to available bandwidth following an idle period[70]. We do not model TCP dynamics as that can complicate the system and reduce clarity in the work.

Every 100ms we recalculate the transfer rate of all individual flows to account for progress of BitTorrent file-sharing. Specifically, the simulator starts with the link j that is most congested i.e., with the lowest per-connection bandwidth. For each connection i using that link j, we subtract the end-to-end bandwidth achieved by i from every link it traverses. This causes a change in the available bandwidth of many links and thereby the bandwidth of connections passing through. Then, we repeat the above calculation by starting with the next most congested link. In the event the transfer rate of a ongoing content transfer is altered, the end time of the piece download is recalculated and rescheduled, thereby ensuring proper dependencies.

6.3.2.2 Modeling BitTorrent Dynamics

We implement the complete BitTorrent protocol exactly according to available specifications, with some minor exceptions. We do not simulate the *endgame mode*, wherein a peer sends multiple requests for the last few sub-pieces to avoid prolonging the download while waiting for content from a slow peer, and *anti-snubbing*, wherein a peer blacklists neighbors that have not provided it content in the last one minute. We believe that these two components address worst case scenarios and are not triggered frequent enough to have load repercussions.

At the BitTorrent layer, the simulation generated peers in each host AS, with the number ranging uniformly between [1, 50]. Each simulation run, on the average, has close to 2500 peers, of which roughly 60% were home users. Then, we create around 50 torrents with their birth time within the past 6 hours or later, and torrent size distributed uniformly between 25.6MB and 2.56 GB. It has been shown in an earlier measurement-based study that the peer arrival process is not Poisson distributed[124]. Hence, we used the trace-driven arrival model proposed in [68], where the peer arrival rate of a torrent follows an exponential decreasing rule with time: $\lambda(t) = \lambda_0 e^{-\frac{t}{\tau}}$, where λ_0 is the initial arrival rate of peers and τ is an attenuation factor to model the gradual drop in popularity of a torrent. We assigned initial arrival rates for each torrent uniformly in the range of $1 - 10$. The attenuation factor was initialized such that around 100 peers arrive in a span of 10 minutes. We assigned each peer to one or more torrents and scheduled its arrival process based on the above mentioned distribution.

We assume that peers do not abort the system when the download is in progress. This is reasonable because the number of peers that abort is a much smaller fraction of peers that arrive and will not alter the effect on the native layer. Once download is complete, we assume that the seed departs after spending the next 1 minute in the system, unless it is the only seed alive for that torrent. The presence of a seed ensures the availability of all pieces. Hence, it is necessary and sufficient for the system to have at least 1 seed.

The other BitTorrent protocol parameters were set at default values (i.e., each peer is assigned a maximum of 50 neighbors initially, each peer unchokes 4 neighbors based on

performance and 1 randomly, each peer recalculates the unchoke list every 10 secs and the optimistic unchoke every 30 secs). The time the unchoking process begins varies from peer to peer. During each run, we randomly pick a stub AS as the focus point and present the performance achieved by peers within it when that AS implements a certain strategy.

6.3.3 Simulation Results

In one of the runs, we picked an AS with two access links with background traffic that was around 20% of the link capacity. The focus AS initially reaches 208 different peers through one access link and 243 peers through the other access link, thereby indicating no initial bias. In Fig. 42, we plot the maximum utilization across all access links once the BitTorrent file distribution is underway. From this figure we observe that the BitTorrent protocol causes large fluctuations in load usage in small time scale (relative to time scale of native layer load-balancing efforts). Furthermore, we see almost full-utilization (0.993) when new torrents are being born, around $t = 5000$ secs and $t = 7000$ secs (not seen in figure), indicating the highly disruptive nature of flashcrowds. In the rest of the chapter, we do not present results for all load scenarios, although we do observe similar behavior. We will describe any deviations as and when applicable.

We also experimented with certain variants of the BitTorrent protocol, none of which satisfactorily reduced the maximum utilization incurred. We summarize them here to augment our understanding of the cross-layer interaction. We performed the following experiments:

- Removing the TFT mechanism, which leads to performance-awareness, and unchoking peers randomly: This caused nearly 10% reduction in the maximum utilization while incurring a finish time inflation of 1.54. Hence, this is not a viable solution for resolving cross-layer conflict.

- Removing rarest piece first: Interestingly, this caused 10.3% increase in the inter-AS traffic without much change in the maximum utilization. There was a minor finish time inflation of 1.12. These indicate that the piece replication was sub-optimal, causing more redundant downloads from each peer (particularly seeds). Thus, rarest piece first policy has a benign effect on TE.

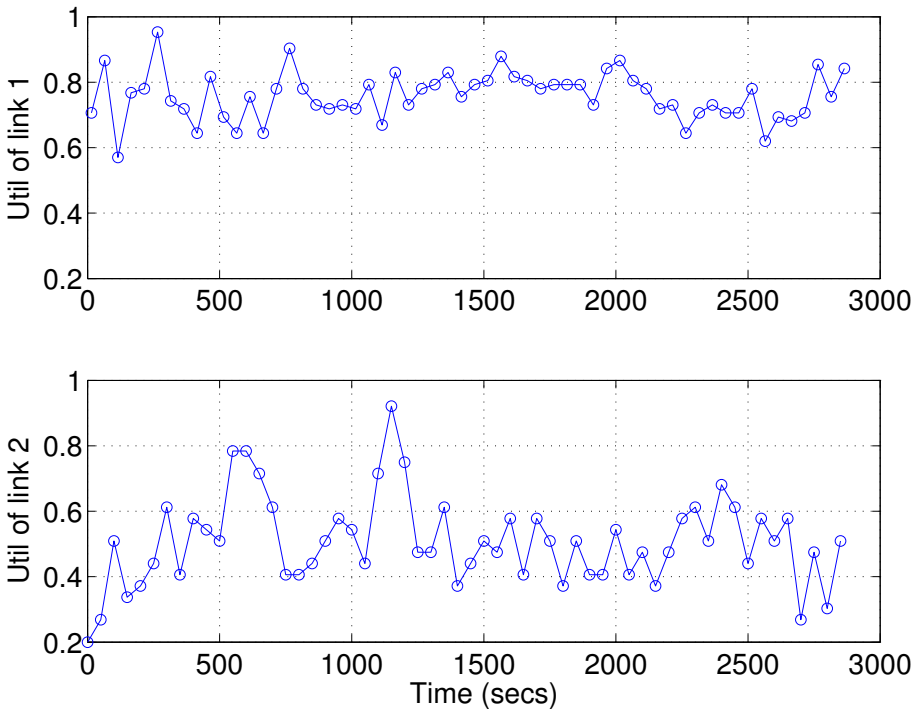


Figure 42: The progression of max utilization without any traffic management.

- Perform optimistic unchoking only for 10 secs in every 30 secs window in order to reduce the duration of altruistic uploading: This causes a finish time inflation of 1.37 without any substantial change in peer loads, indicating that the full 30 secs of optimistic uploading is typically required to improve a peer's standing among its neighbors.

- Freeze list of unchoked peers 10 minutes after arrival: We did not see substantial reduction in peak load because the optimistic unchoke alone was often sufficient to sustain the piece exchanges. The finish time inflation incurred was 1.20.

- Tuning the different unchoking timers: This did not cause any consistent change in the utilization value or finish times.

6.4 Existing Native Layer Strategies

In this section we present the consequences, if any, when the native layer intervenes to reduce the cost incurred by adopting one of the following three strategies:

6.4.1 Effect of Inter-domain Traffic Engineering

TE is a crucial procedure in modern ISP networks to balance load and remove bottlenecks. Typically, it uses a particular snapshot of the traffic demand matrix to derive the set of routes that achieve a specific load-balancing objective. The frequency of re-engineering the routes depends on the amplitude of change in the traffic matrix or the desired periodicity. This operation can be performed both on inter-domain links and on intra-domain links. The behavior and the methodology vastly differ with both flavors. In this thesis, we only focus on the inter-domain effects of BitTorrent file-distribution.

There are many moments when the volume of traffic through a particular ingress or egress inter-domain link is substantially higher than that seen in all other links. This is typically what inter-domain TE targets and aims to balance. It is also possible that an AS will want to reduce the overall cost incurred, when the individual inter-domain links have certain weights (often to reflect the monetary cost incurred). In our work, we study the effects of adopting two forms of inter-domain traffic engineering, *egress TE* and *ingress TE*, each with network-specific objectives.

We perform egress TE to control how the traffic leaves an AS. Since an AS controls the decision process on its own outgoing BGP routes, this can be accurately achieved at a fine granularity. This is easily architected by choosing the appropriate route to use for reaching a particular destination prefix through its adjacent ASes. The common ways to force traffic for a prefix to leave the network through a certain gateway is to use a combination of `local_pref` attribute and IGP weights to include the choice of the appropriate border router.

On the other hand, ingress TE helps control the flow of traffic into the AS. This can be achieved through a combination of *selective advertisements*, wherein an AS advertises certain prefixes through a particular inter-domain link, and AS-path prepending, wherein an AS artificially concatenates more AS numbers on the advertised path so as to inflate the AS path length attribute. These strategies work at a coarser granularity because it aims to influence the decision of ASes a few hops beyond their adjacent ASes.

Each form of TE can work at two different modes: i) *Offline*, when the native network

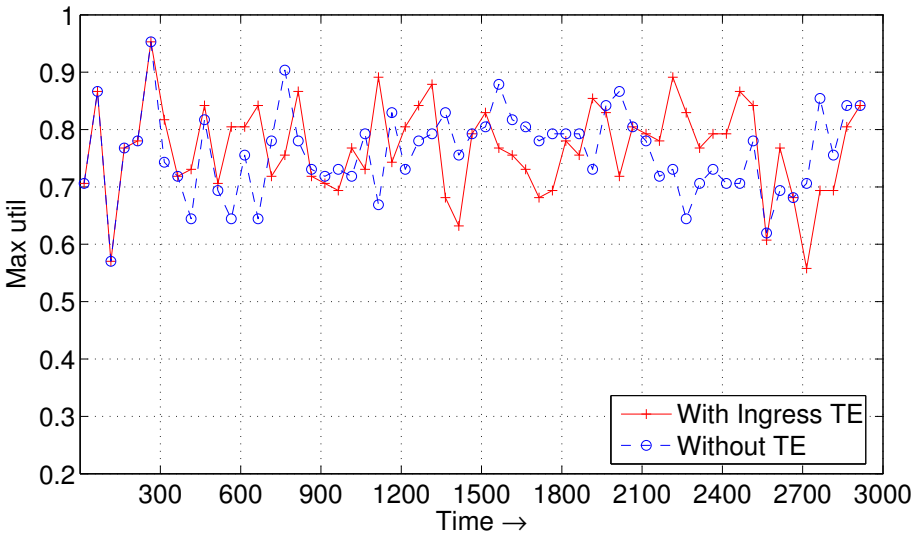


Figure 43: The progression of max utilization with and without ingress TE. TE was performed every $\delta = 300$ secs, starting at $t = 300$ secs

uses previously logged traffic demand matrix to predict the future demands and engineers the routes to balance the predicted load, ii) *Online*, when the native network uses dynamically evolving traffic demand matrix to rapidly alter the routes and balance the load. Refer to [46] for a more complete survey of inter-domain traffic engineering solutions.

In this chapter, we focus on ingress and egress TE that attempts to solve the following optimization problem: $\min\{\max_{a \in E} C_a \times \mathcal{U}_a\}$, where E represents the set of inter-domain access links, $\mathcal{U}_a$ is the utilization of that link and C_a presents the weight associated with that link. The optimal solution will determine the amount of traffic associated with each source or destination AS, and the appropriate route from/to that AS so as to distribute the load evenly across all possible providers. We understand that the reality in inter-domain TE solutions cannot achieve this level of granularity. However, we still adopt this model in our simulation in order to get an idea of how disruptive the BitTorrent protocol can be, even in the optimal TE setting.

It is possible that the objective of TE is to minimize the total weighted cost incurred, rather than the maximum value, or the unfairness in load distribution across the access links. However, we still adopt the minimization objective as it achieves a good mix between reducing the total cost and the unfairness in load distribution. We simulate only the online

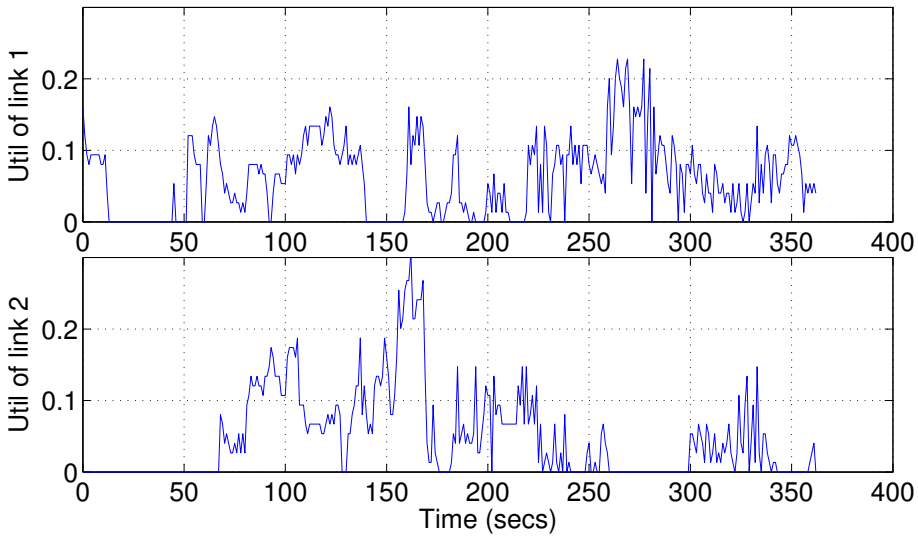


Figure 44: The utilization of link capacity caused by one peer in the focus AS.

mode of TE, although we expect our results to be equally applicable to the offline mode.

In Fig. 43, we plot the utilization seen by each link once the BitTorrent file distribution is underway and online ingress TE is performed by the focus AS every 5 minutes. We observe that TE manages to bring down the maximum link utilization for a few seconds, after which the BitTorrent protocol drives the value back up, thereby indicating its disruptive nature. We also plot the progression of maximum utilization value for the base scenario without TE. From this we observe that applying TE did not make much of a difference on the peak load variations. Further, we did not observe any change in the average finish time. This indicates that ASes carrying large volumes of BitTorrent traffic will not benefit from deploying conventional TE schemes.

We plot, in Fig. 44, the link utilization of one particular peer that resides in the focus AS to learn about individual client behavior, when ingress TE is implemented. We observe that the peer's downloading decision works at a fast time scale relative to TE and tends to maximize the utilization of available link capacity frequently. The combination of many such peers spread across multiple co-existing torrents cause the frequent disruption of the TE objective.

Table 15: Summary of ingress performance achieved by locality-based traffic management strategies

Strategy	% downloaded internally	Max util	Avg. finish time inflation
Basic BT	13.55	0.993	1
Biasing peer set	28.28	0.8173	1.5091
BT Request Caching	53.82	0.7606	1.1971
Caching + Biasing	58.39	0.5968	1.3030

6.4.2 Effect of Locality-based Traffic Management

Typically, the neighbors of a host may be spread over multiple domains based on the geographical diversity and the period in the lifetime of the torrent. This reflects in a large volume of inter-AS traffic. The general idea of locality-based traffic management is to exploit the replication of content within the AS before reaching out to external peers. This idea of promoting locality in BitTorrent has been proposed by researchers[80, 50], as well as vendors[21, 134]. The main objective is to reduce the inter-AS traffic without impacting the user experience. Furthermore, these strategies do not require any change to the original tracker or client.

Locality can be implemented in the AS by deploying proxy-trackers, which can promiscuously intercept the outgoing bootstrapping requests from new peers or neighbor set requests from existing peers. The proxy tracker now returns N peers, with k peers chosen within the local AS. It has been shown that having a small subset of neighbors from outside the AS helps ensure TFT incentive and maintains sufficient availability of content within the AS[50]. As intra-domain resources are not as limited as inter-domain resources, the peers do not suffer much inflation in the download time. Another form of locality that an AS can implement is the caching (storing) of previous requests that were propagated to external peers. Caching can help reduce redundant requests from going to external peers and can serve same requests from other leechers immediately.

We implemented both forms of locality-based traffic management in our simulation framework, for the same focus AS. With biased neighbor selection, the number of local peers returned was the minimum of either the number of peers within the local AS attached

to that particular torrent, or 25 (value of k). With request caching, we stored a maximum of 10,000 pieces in order to serve future requests. When full, the cache discards the least recently used items first. Table 15 presents the results we observed with these two strategies and their combination. We see that biasing the neighbor set reduces the amount of inter-AS traffic and the maximum utilization across both links. However, the finish time of the peers within the AS noticeably increased. This is because the peer selection has reduced the amount of content offered to external AS, thereby harming the TFT incentive and making it longer to obtain newer content. Thus, it took longer to propagate newer content within the AS with biased peer selection. With caching, we observe that the inter-AS load (volume and the maximum utilization on each link) has substantially reduced, without causing a drastic change in the finish time. This is because the peers still upload content to external peers and retain the TFT incentive. However, the seeds will favor less a peer behind a caching system because the peer downloads very little actual content from outside. This explains the slight inflation in the finish time. When we deploy both locality-aware strategies, we see the best performance with regards to inter-AS load. The finish time is in between that of biased neighbor selection and caching as can be expected.

The egress performance was not substantially different from that observed for the base scenario (without locality-aware strategies). We noticed that the amount of traffic egressing (uploaded from) the focus AS is least for biased neighbor selection strategy with about a 3% reduction, while that of caching strategy showed negligible deviation from the base scenario. Thus, the strategies typically improve only the ingress performance.

6.4.3 Effect of Bandwidth Throttling

In an effort to reduce network traffic, and consequently incurred costs, some ISPs throttle or rate limit the aggregate BitTorrent traffic. This way the overall BitTorrent traffic is kept within fixed limits and does not disrupt TE. Clearly, this approach requires lesser infrastructure cost than that needed by locality-based traffic management strategies. The popular BitTorrent vendor Azureus now maintains a survey of ISPs that are known to limit the bandwidth consumed by BitTorrent applications[8], thereby indicating the popularity

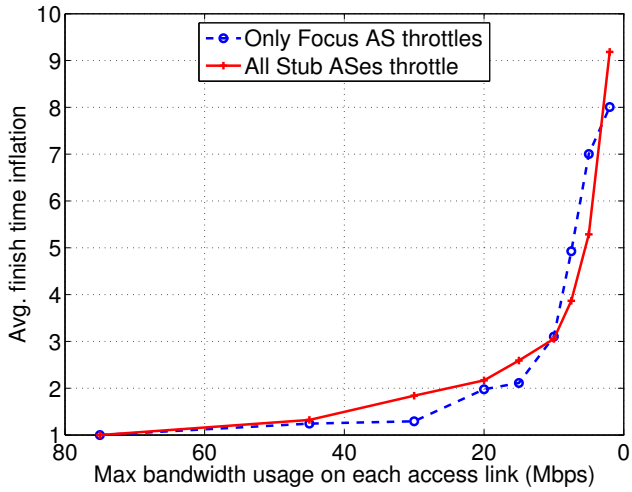


Figure 45: The penalty incurred when ASes throttle aggregate BitTorrent bandwidth using traffic shaping.

of this approach. We investigate this as a possible strategy in the multi-layer interaction and obtain insights on its effectiveness. We present in Fig. 45 the results observed when the focus AS performs bandwidth throttling and when all stub ASes perform it. We observe that the user experience degrades substantially as the level of throttling increases. We see a non-linear behavior wherein the degradation increases steeply as the throttling rate crosses a certain knee-point. This is because at that high level of throttling the bottleneck in the BitTorrent communication is almost always the AS enforcing the bandwidth limitation. There are certain points when the average finish time inflation is higher for the peers in the focus AS than for peers in all stub ASes. This is because at those points the throttling makes the upload/download rate of peers in the focus AS look relatively unattractive to other external peers, causing it to lose the TFT incentive and incur a higher inflation in finish time.

The results in the figure correspond to the case where throttling is enforced in both directions. When we enforce throttling only in one direction, the result is dependent on the composition of the AS (number of leechers and peers) and the bandwidth characteristics of the neighbors' AS. However, the effect is not symmetric. It hurts the user experience more when the download rate is throttled, as opposed to when the upload rate is throttled. In our simulation, throttling the download rate to 5Mbps yielded an average finish time penalty of

5.752, while throttling the upload bandwidth to that rate yielded only a penalty of 1.114. This corroborates the observation that altruism in the BitTorrent system ensures even low bandwidth peers get to download at a rate more than they upload to the network, as long as there are other high bandwidth peers present[14]. Thus, upload bandwidth throttling at a few ASes does not seem to affect the finish time.

6.5 Friendly BitTorrent Strategies

The BitTorrent protocol and traffic engineering are two non-cooperative entities participating in a repeated game. This causes them to constantly retaliate to the other player's strategy. The previous section presented the downside of bandwidth throttling to peers within the AS. In retaliation, recently, many BitTorrent client vendors collaborated to create an end-to-end encryption strategy called Protocol Encryption (PE), which helps avoid the traffic shaping software[20]. This form of interaction will continue indefinitely as the two layers compete for limited substrate resources. In an effort to end this cross-layer conflict, we propose some "friendly" strategies that BitTorrent peers can use to reach a mutually agreeable operating point. Our objective is not to completely dampen the load fluctuations, but rather to prevent BitTorrent applications from depleting resources.

Clearly, the locality-based neighbor selection investigated in the previous section may be supported by the tracker itself without relying on extra infrastructure from the individual ASes. In our simulation, we observe in our focus AS that almost 33% of the content is downloaded internally when all torrents support locality globally, in comparison to 28% when enforced for only peers within the focus AS and 22% when only one popular torrent supports locality globally. Although this decrease in inter-AS traffic is attractive, this option does not provide a guarantee or a minimum bound on the reduction in inter-AS traffic achieved, since the level of locality itself may vary with the particular torrent. This motivates us to derive more consistent strategies.

In this section, we propose two new strategies of which the first one does not require a change to the BitTorrent protocol and can be enforced at respective client applications, while the second one requires the addition of a new primitive in the inter-client communication.

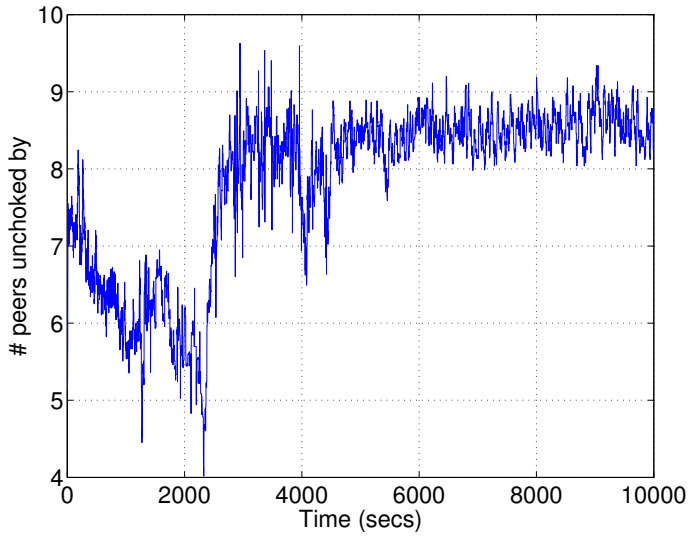


Figure 46: The average number of unchokes seen during each second of the simulation.

6.5.1 Limiting Number of parallel Downloads

The main reason for the heavy surge in BitTorrent traffic is that the unchoking process does not limit the number of peers that might unchoke a particular host. In the event a particular host is simultaneously unchoked by a large number of peers, it gives an opportunity for this host to act selfish and increase the number of parallel downloads. According to the BitTorrent protocol each node unchokes a maximum of 5 peers (including the optimistic unchoke). Thus, if there are N peers associated with a torrent T there will never be more than $5 \times N$ unchokes issued overall. Yet, our investigation of the average number of unchokes (representing the average number of hosts ready to offer content) received by all peers in the BitTorrent system indicates that the average is much higher than the ideal scenario. We observe from the simulation results, presented in Fig. 46, that the average lies between 8–9 peers. The reason for this large number of outstanding unchokes is that:

- the unchoking decision is not communicated across peers. It is also possible that the neighbor set of a peer A's neighbor will vary drastically from that of peer A.
- the timeline of the unchoking process is not synchronized between each peer, causing moments when multiple peers are attracted to the same peer A's upload rate and unchoke it.

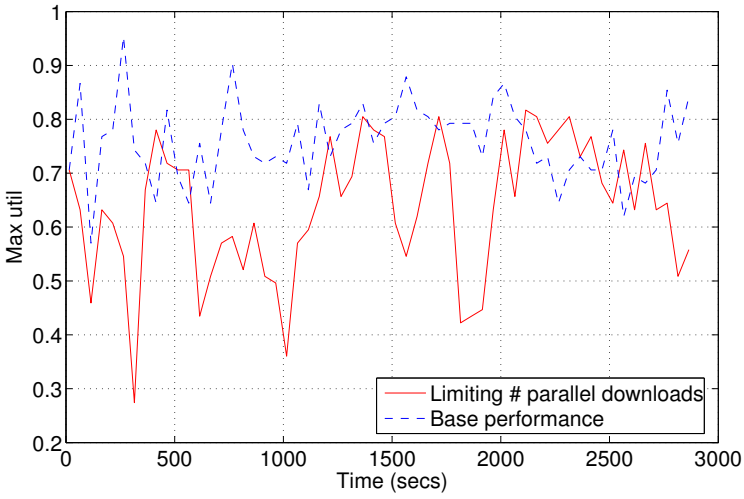


Figure 47: The maximum utilization seen when peers limit number of parallel downloads to 5.

- the block availability exhibits high variability, causing moments when fewer neighbors of a peer A are interested in the content available in peer A. This reduces the contention for unchoking and makes the same peer B receive multiple unchoke signals.

- the torrent has a healthy number of seeds present. Seeds will never be unchoked. Thus, $5 \times N$ unchokes will be distributed over $N - s$ peers. where s represents the number of seeds belonging to that torrent.

We restricted the number of parallel downloads a host performs to 5, for all hosts in the system and noticed a significant reduction in the maximum ingress utilization from 0.94 on the focus AS to 0.852, during the 3000 secs we inspected. Fig. 47 shows the progression of the maximum utilization of ingress links when we enforce the limitation. We observe that the maximum utilization is substantially lower than the basic system and also that the general load fluctuations has also reduced. Furthermore, this caused an overall inflation in the finish time of only 1.1501. Hence, this is an attractive solution to reduce the level of contention by way of tuning the BitTorrent protocol. A similar voluntary restriction of the upload rate can also help the system without causing a substantial increase in the download time. Furthermore, reducing the number of parallel downloads/uploads, can improve TCP performance[110].

6.5.2 Avoiding Common Neighbors

Next we propose a simple improvement to the BitTorrent protocol which performs bilateral negotiations with each of its neighbors to learn about the peers two hops away. Specifically, our addition aims to determine if each of its neighbors is also a neighbor of another peer in the same AS. This basic knowledge can help reduce overloading certain access links, as the peak loads are often caused when the same uplink is chosen for multiple connections. If a particular neighbor is common between two peers in the same AS, then we use some randomization in picking this neighbor for any data transfer i.e., we choose this neighbor for unchoking or download at a probability p. Thus, randomization serves as a form of load-balancing (similar to Valiant load-balancing[159]). Furthermore, this solution does not require the tracker to retain state information about the neighbors it assigned to prior peers requests. A good value of the probability p must depend on the number of neighbors within an AS that share the common neighbor. We can have the negotiations return this value as well.

We plot, in Fig. 48, the performance seen by the focus AS when this strategy is deployed in all BitTorrent peers. We observe that the maximum utilization of ingress links over the 3000 secs plotted is 0.85 while that incurred by the basic system is 0.94. This indicates a substantial improvement in the available bandwidth of those links. Furthermore, we observe that the general trend of maximum utilization is significantly lower than the basic systems. This modification caused an overall finish time inflation of 1.187, which is generally acceptable.

If each peer in the system had global information, it would prefer to have a large fraction of neighbors from within its AS, in order to bypass the bottlenecked access links, and keep a small fraction of external neighbors that are unique to itself, in order to obtain rare content it can offer neighbors within its AS. This implies that the best strategy would be a combination of biased peer selection and avoiding common neighbors.

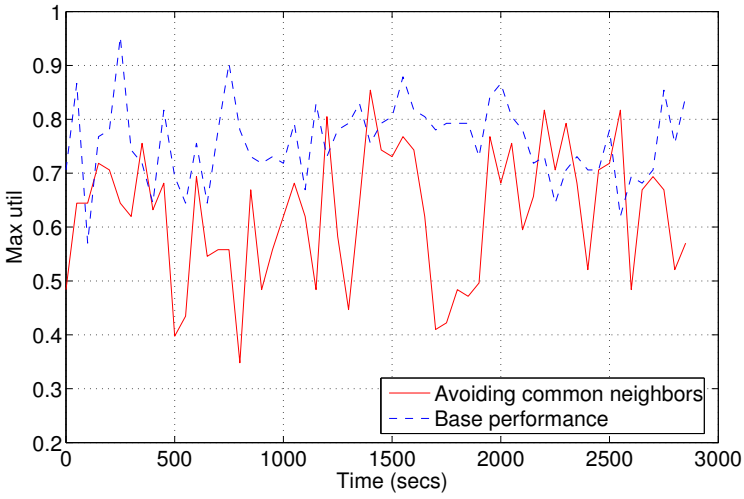


Figure 48: The maximum utilization seen when peers randomly avoid common neighbors.

6.6 Related Work

A number of previous studies have investigated the BitTorrent protocol analytically[170, 125], through simulations[14, 15] and through measurements-based analysis[47, 68, 123, 94]. We use the inferences from each of these studies to fine tune the model we used in our simulation and for proposing efficient strategies that causes least disruption of TE. Hence, each of these studies are orthogonal to our work as we investigate the overall "BitTorrent effect" on the native layer. In particular, we adopt the peer evolution model proposed by Guo et. al[68] in our work as it seems to represent the closest model to reality and has been corroborated with evidence from traces obtained from large commercial server farm and a group of home users.

6.7 Summary

In this chapter, we investigate the multi-layer interaction between BitTorrent file-sharing applications and inter-domain traffic management strategies. Using real traces obtained at the Georgia Tech border router and a discrete-event based simulation, we show that the performance-awareness of BitTorrent protocol causes it to disrupt any load-balancing efforts of the inter-domain TE. To our knowledge, our work is the first to investigate the interaction of overlay applications with inter-domain TE. Further, we observe that existing strategies like locality-based traffic management and bandwidth throttling, that are quite

effective in managing BitTorrent traffic, are either too expensive for an AS to deploy or leads to deterioration in user experience. To resolve this, we propose and investigate two BitTorrent strategies to alleviate the detrimental effects of this cross-layer contention and steer interaction towards a mutually agreeable point for the two entities. Specifically, we propose limiting the number of parallel downloads and randomly avoiding neighbors common to peers in the same AS. We show that these two strategies perform satisfactorily for both entities. Our simulation-based evaluation primarily offers a proof-of-concept before evaluating it on an actual testbed. In the current Internet, we see a constant power struggle between the native layer and the selfish overlay layer. As the level of BitTorrent usage surges, there is a clear need for our strategies, in order to prevent wide-spread "bandwidth war".

CHAPTER VII

OVERLAY-FRIENDLY NATIVE NETWORKS: A FRAMEWORK FOR CROSS-LAYER COOPERATION

7.1 Introduction

The design of today's IP networks calls for the internal network elements to concentrate on the simple forwarding function, leaving the high-level functions to the end-points. In accordance with that design approach, overlay networks have emerged as an effective way to implement functionality which would otherwise require significant change at the native IP layer. These overlay networks are constantly evolving to address the increasing demand for new services, while the native network has been allowed to stay unchanged[1]. Such an approach can overburden the overlay networks themselves as they must assume the full responsibility for any functions or services beyond best-effort unicast forwarding. Further, even if current overlay networks can perform the services expected from them, the amount of achievable performance gain is limited due to this inflexible design approach. To overcome these long-term problems, we suggest the native network evolve gradually to suit the overlay network and the new services sought. Hence, we investigate ways to improve the performance by *modifying* the operation of the current native layer.

In particular, the performance of the overlay network services is restricted because of unawareness of the native layer information and a lack of control over the native layer's decisions. Considering those reasons, there is a wide spectrum of optimizations one can propose for the overlay services, as indicated in Fig. 49. In the figure, design choices that are furthest away from the origin tend to provide the most benefit. Past research on improving overlay services has focused predominantly in two dimensions namely *controlling overlay layer* and *knowledge of native layer*. Examples include work on tuning the overlay

[1]The overlay does not expect any help from the native layer because it cannot be modified and the native does not provide any help because no overlay requires it to - an instance of the classic chicken-and-egg problem.

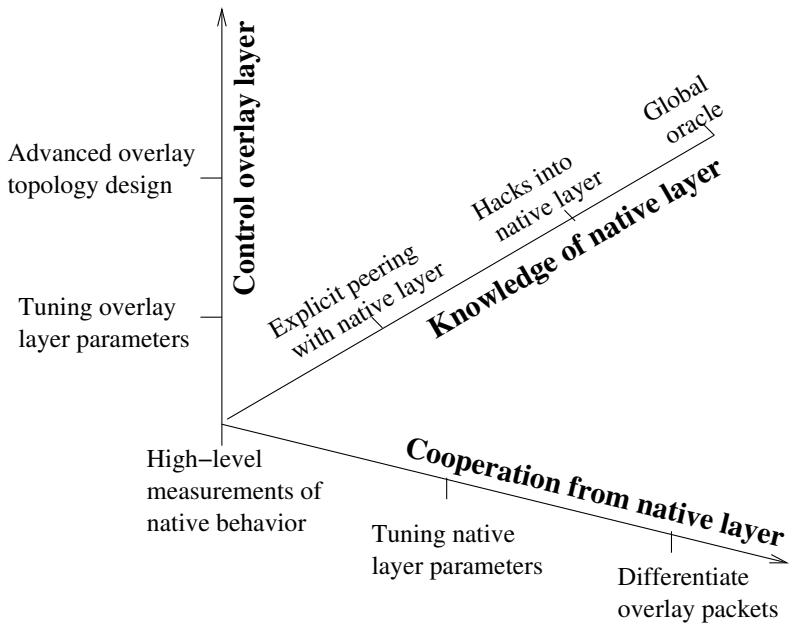


Figure 49: The various degrees of freedom available to overlay service developers.

layer parameters ([6, 138]), advanced overlay topology design ([69, 142, 95]), and obtaining native layer information ([175, 114, 24]). We explore a new degree of freedom that involves *cooperation from the native layer* for improving overlay services. The solutions in this direction are generally classified as providing an *overlay-friendly native network* (OFNN), briefly defined as a native network that caters to the overlay applications without compromising on the performance of the non-overlay applications. Such a native network should also be easily deployable, backward compatible and inexpensive to manage and operate.

The concept of an OFNN, however essential, comes across as a *contradiction in terms*. This is because overlay networks were conceived to obtain new network functionality without modification of the underlying native network. If it were feasible to modify the native network, the need for the overlay application is obviated. Therefore, modification of the native network to suit the overlay application seems to be a contradiction to the purpose. However, this is not the case always. There are modes of change in the OFNN which we argue does not constitute such a contradiction. For example, consider a certain feature that is currently implemented by means of an overlay and whose performance we would like to improve. Making a distinction that native layer operations are carried out by programmed

functions which receive *parameters* (from managers or other functions) to guide their operation, we are faced with several choices to better support the feature at the overlay layer. Among these are:

A. Add a new function to the native layer.

B. Modify the existing functions in the native layer.

C. Tune the native layer parameters, without altering the functions at the native layer.

Clearly, the solutions put forward by options A and B are invasive procedures, requiring alteration of the native layer code. Hence, we do not see them as a feasible solution. In this thesis, we advocate the use of option C, where we do not need to rebuild the native network and the functions used at the native layer remain fundamentally the same. We believe that this native layer tuning is a pragmatic approach to constructing an OFNN.

From the discussion above, we can see that some OFNN approaches represent a contradiction and some do not. Based on this observation, we classify the OFNN approaches into two categories - *contradictory* and *non-contradictory*. Furthermore, we present native layer tuning as a non-contradictory OFNN approach and illustrate the overlay-friendly tuning of the native layer BGP multi-exit discriminator (MED) attribute and IGP cost.

Recent work on network virtualization highlighted the importance of addressing the impasse in progress of the current Internet[141, 158, 90, 37]. It suggests two perspectives in targeting the problem - that of a purist who considers that the overlay network is a tool to experiment a feature before full-fledged deployment in the native network, and a pluralist who considers that the diversity brought about by overlay networks should become a fundamental part of the native network. In this thesis, we advocate an intermediate viewpoint where we consider the need for overlay networks to be inevitable, in addition to requiring some minor alteration of the native network. The discussion in the rest of the chapter is merely a first step in that direction and is not intended to be a full-fledged analysis like the earlier chapters.

The remainder of this chapter is organized as follows. We elaborate on the definition and design goals of the OFNN in Section 7.2. Some examples of the contradictory OFNN

approach are briefly described in Section 7.3. We present our novel approach towards native layer tuning and identify some of the parameters that can be tuned in Section 7.4. Section 7.5 summarizes our position and suggests the required future work.

7.2 Overlay-Friendly Native Network

We motivated the need for the overlay-friendly native network in Section 7.1. This section presents the design in better detail.

7.2.1 Definition and Design Goals

We define an *overlay-friendly native network* as a native network that incorporates special changes targeted towards the benefit of the overlay applications, without causing a negative effect on the performance of the non-overlay applications. We use the term *friendly* to highlight the special treatment rendered to the overlay application, irrespective of whether the native layer is aware of its existence. The change incorporated may not help all services, but is rather an optimization on a case-by-case basis. We refer to some changes as *invasive* to indicate whether the native layer functions need to be altered (reprogrammed).

The following are some of the design criteria mandated by the framework:

- The foremost concern is that adjusting the native layer must yield a significant performance gain for the overlay application. The performance metric depends on the type of service provided by the overlay network. For example, resiliency services are metered by recovery time, and multicast services by stress factor.

- It should have no negative effects on the way non-overlay applications and their traffic are handled. For instance, we should take special care that a change made in the native layer to support special packet scheduling of the overlay traffic must not lead to starvation of non-overlay traffic. In the presence of multiple coexisting overlay applications, the modification should have no negative effects on the traffic of the other overlay applications.

- The alteration made to the native layer must be conservative i.e., it should generate negligible, if not zero, extra overhead (in terms of processing, protocol, memory, and

other resources) and must not cause a deterioration in existing performance of the overlay traffic (in terms of the classic metrics like throughput, forwarding rate, latency, recovery time).

- It should facilitate a practical roadmap for wide-spread deployment. In cases where the change to the native network is invasive, the benefits should gradually accrue as more native routers are altered to support the feature at the overlay layer. Such a change is considered incremental. To satisfy this requirement, the overlay network must provide minimal expected services even when the change is unavailable or partly available in the native network.

- The native network must be backward compatible in that it should still process legacy overlay applications (without the new functionality) in the expected manner. For example, consider a change in the native layer which aids in overlay path diversity (minimal overlap of the native route used by the overlay links). An overlay application that is incapable of using this added support from the native layer must still receive the basic connectivity between its overlay nodes.

From the design criteria stated above, we note that one key design goal is to avoid negative impact on non-overlay traffic. In some cases, the change made to the native layer might even be beneficial to certain non-overlay applications. To provide these potential benefits to non-overlay traffic, the OFNN should not distinguish between the different applications sharing the native network, wherever possible.

We also infer from the design criteria above that the change incorporated in the native layer does not necessarily depend on the proportion of native traffic that belongs to the overlay layer. However, certain invasive changes to the native layer become more justifiable in the presence of a high volume of overlay traffic.

7.2.2 Implementation Options

Having established the basic design goals of an OFNN, we proceed to list some of the possible design choices available, on top of the sample set listed in Section 7.1, for modifying the

native layer[2]:

A. Add a new function to the native layer.

B. Modify the existing functions in the native layer.

C. Tune the native layer parameters, without altering the functions at the native layer.

D. Create packet level filters at strategic points of the native network to identify the overlay traffic or to provide functionality that can only be exploited by the overlay traffic. Once identified, the native layer can render some special treatment to these packets. This serves as a form of differentiated service.

E. Use a particular native layer function in a different manner, in combination with overlay-specific hacks. For example, the usage of native IP multicast for propagating queries in P2P networks, in association with reserving a permanent IP multicast address for each P2P network, is yet another approach for building an OFNN.

F. Make the native layer subsume the overlay feature i.e., add the whole feature that is currently being offered at the overlay layer to the native layer.

We can see that options A, D and E are inherently backward compatible by not affecting the set of functions currently in place. They treat all legacy traffic (both overlay and non-overlay) in the same manner as before. On the other hand, options B and C need special care that this change does not negatively impact the non-overlay applications. Option C is the only scheme that does not require a change to the native layer. It is interesting to note that option F obviates the need for the overlay application. Hence, it represents the highest level of modification at the native layer.

7.2.3 The Contradiction

We briefly explained the presence of a contradiction in Section 7.1. The contradiction arises in the OFNN when the overlay network, which was conceived to avoid modification of the

[2]This list is by no means comprehensive.

native layer for obtaining new network functionality, demands a change in the native layer to improve its performance.

We would like to add that the contradiction arises because the overlay service is unable to provide the best performance autonomously. This inability can be attributed to the fact that overlay nodes are limited in number, mostly located at the edge of the network and are basically users of the native network services. For example, consider the following problems in previously proposed overlay services:

- Resilient overlays[6] - suffer from high chance of irresolvable overlay network partition, owing to lack of control over path diversity.
- Multicast overlays[28] - suffer from high stress and stretch, relative to native multicast.
- QoS overlays[149] - suffer from lack of absolute service guarantee, owing to presence of other non-conforming non-overlay traffic.

These problems motivate the need for support from the native layer, thereby leading to the contradiction.

We argue that some of the implementation options in Section 7.2.2 do actually represent a contradiction. Hence, based on the type of change, we classify the approaches adopted into the following two categories:

- *Contradictory OFNN* - when the required change is invasive, leading to alteration of the native layer functions. Options A,B,D,E and F in Section 7.2.2 represent this approach. We present existing proposals for these OFNNs in Section 7.3.
- *Non-Contradictory OFNN* - when the required change is non-invasive. Option C in Section 7.2.2 represents this approach. We present some instances of this OFNN in Section 7.4.

7.3 Examples of Contradictory OFNN

The following are some OFNN proposals that require modification of the native layer to bring about a performance improvement for the overlay service.

7.3.1 Resource Sharing

One theme in past work on overlay network design is the addition of an intermediate layer to interface between the native and overlay layers. This new layer helps improve the performance of the overlays without exercising any control over the native layer. As suggested by Fig. 49, these schemes aid overlay services by obtaining a higher level of information (about topology, routes, and resources) from the native layer. However, there are obstacles to achieving that. The Internet has attained its current level of scalability mainly by information hiding. Hence, there is a fundamental limit on the amount of information one can obtain, which current probing schemes cannot subvert. Therefore, obtaining privileged information requires an invasive change at the native layer.

The work on network virtualization[141], diversified Internet[158, 90], Opus[18], routing underlay[114], service-oriented Internet[24], and Brocade[175] propose using this intermediate resource provisioning layer to allocate resources to the different overlay networks on top. In certain cases, this intermediate layer serves as a repository for routing services, high-level performance measurements and BGP information. It can also be used to provide optimized routes between peers by exploiting knowledge of underlying network characteristics.

All solutions listed above, albeit better than overlay networks that operate completely independent of the native network, are still plagued with the problem of limited gain (See Section 7.2.3 for details). Hence, we argue that greater performance gains can be obtained only when the functioning of the native layer is altered to bring about a synergy with the overlay layer.

7.3.2 Network Support for Overlay Networks

Jannotti proposed two new primitives for implementation in the native layer - *packet reflection* and *path painting*[77] - to provide advanced routing services like packet duplication and route examination. These two primitives support the efficient construction and operation of certain overlay networks. The work also showed ways to incrementally deploy the primitives in the Internet. However, this modification of the native network is invasive and inhibits widespread deployment.

7.3.3 Active Networks

Active Networking is an architecture proposed to address the problems of network stagnation[153, 126]. It suggests enhancing the network elements with more processing abilities, so that the applications at the end user can obtain higher benefits by uploading new protocols and code to the intermediate routers. This results in a higher programmability of the native layer. One can see that the active network is very much an OFNN, where the native layer helps enhance the performance of the overlay applications by being programmable. However, the active networking approach is highly complex and needs significant modification of the native layer.

7.4 Example of Non-Contradictory OFNN: Tuning Native Layer Parameters

The current native layer parameters are tuned for the existing users of the native network services. However, there are parameters which can be tuned in an overlay-friendly manner without any negative impact on other applications. We focus on the tuning of native layer parameters that affect functions like routing (IGP or BGP), scheduling (IP priority), multicast, and security (firewalls, address translators).

In general, network administrators may leave some parameters at the default value configured by the equipment vendor. In such cases, it is usually easier to justify the overlay-friendly tuning. It is perhaps inadvisable to tune parameters that have been purposefully set at a particular value, so as to avoid any conflict with pre-established policies.

In this subsection, we present ways in which the native layer can provide the following functions, noted from our literature survey as a common requirement for many overlay services:

- Earlier failure detection by the native layer[138]
- Symmetric routing for native routes between end hosts[138, 114, 27]
- Coherent cost metrics between the two layers[138, 130, 98, 83]

To offer the above support, we recommend the overlay-friendly tuning of the routing

protocol keepAlive-time, BGP MED attribute and the IGP cost of each link, at the native layer, to aid the overlay applications. We assume that both the native and the overlay layers employ a dynamic routing protocol to adapt to changing network conditions.

A) Tuning the Routing Protocol hello-interval for Faster Detection:

In Section 3.1, we highlighted the fact that a dynamic routing protocol is essential in the overlay layer to significantly enhance the overlay networks survivability. However, in cases where both the native and overlay layer were capable of rerouting around a link failure, the native layer rerouting was shown to be the optimal one in terms of path cost inflation and number of route flaps, as seen from the results in Section 3.5 (See the results for route flaps and path cost inflation of Dual Rerouting when the overlay hold-time is bigger than the native hold-time).

In Dual Rerouting, insuring that recovery will take place at the native layer first can be achieved in multiple ways:

- Using a device or hardware notification to trigger the native rerouting.
- Adding an overlay-to-native signaling protocol to jump-start the native rerouting as soon as the overlay layer detects a failure.
- Setting the native layer's keepAlive-time to a value much smaller than that at the overlay layer.

The first option is not always possible due to limitation of the physical layer. The second option is an invasive procedure that requires alteration of the native layer code to support the new feature. Hence, we do not see it as a feasible solution. The last option of tuning the native layer timer value is the most feasible. We do not need to rebuild the native network and the functions used remain fundamentally the same. However, we need to adhere to the following two constraints while tuning the keepAlive-timers:

1. The tuning should not generate any extra overhead.

2. The effective detection time (defined earlier as the smaller of the detection times at either layer) should be the same.

The keepAlive-time for each layer is typically chosen based on the amount of *keepAlive* message overhead the layer can deal with[97, 4] and the amount of hit-time the system can tolerate in the event of a failure. This indicates a tradeoff that exists between the responsiveness to a failure and the protocol overhead. We define the routing protocol overhead as the number of *keepAlive* packets sent per second on the link under consideration. The sum of protocol overhead in each native link, contributed by both the layers, represents the overall protocol overhead. The overall protocol overhead is calculated as:

$$\text{Overall protocol overhead} = \frac{\text{Number of native links}}{\text{Native keepAlive-time}} + \sum_{\text{Overlay link}} \left(\frac{\text{Native hop count of the overlay link}}{\text{Overlay keepAlive-time}}\right) \quad (1)$$

It is widely believed that the overlay network can reroute traffic around native failures earlier than the native network[6]. The main reason is that the native layer recovery time can be long due to the reduced periodicity of *keepAlive* messages to conserve native routing protocol overhead[4].

This earlier recovery at the overlay can be achieved by setting the keepAlive-time to be relatively short. The additional overlay routing protocol overhead incurred is considered to be manageable because the overlay network tends to have fewer nodes and links. However, this is a flawed argument. This can be seen by examining our estimates of the routing protocol overhead at the two layers. We noticed that the overhead incurred at the native and overlay layers are quite comparable. This can be attributed to the following three factors:

- Each overlay link is comprised of multiple native links.
- The overlay network has higher degree of link connectivity than the native network.
- There can be multiple coexisting overlay networks on the same native topology.

The significance of the above reasons can be verified from the expression for overall routing protocol overhead in Equation (1). The fact that the protocol overhead in the two layers are comparable leads us to revisit the choices of native and overlay keepAlive-time.

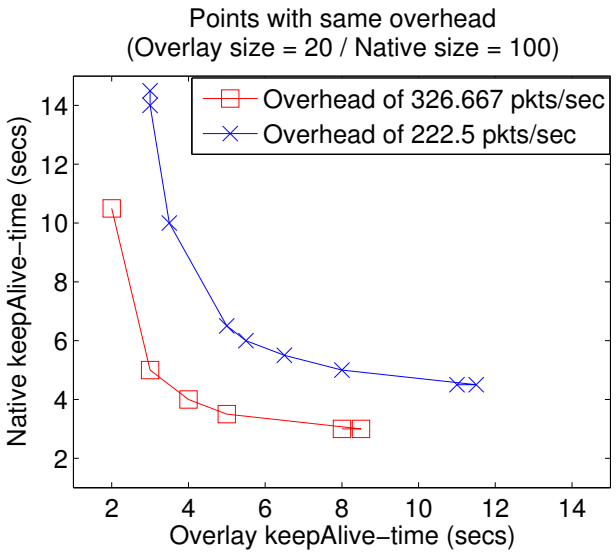


Figure 50: Possible choices for keepAlive-times when protocol overhead is same.

We consider the effect of decreasing the native layer keepAlive-time parameter. This can cause an increase in the overall routing protocol overhead. We offset that by increasing the overlay layer keepAlive-time value, which ultimately keeps the overall routing protocol overhead for both the overlay and native network under control. Maintaining the protocol overhead at the same value is possible because the overall protocol overhead can be the same for multiple pairs of native and overlay keepAlive-time. Fig. 50 shows such tuples for a particular topology with 20 overlay nodes and 100 native nodes. Each point on the curve gives us the keepAlive-time setting for the two layers (X-coordinate value for overlay layer and Y-coordinate value for native layer) so that the desired overall routing protocol overhead can be obtained.

Fig. 51 presents multiple scenarios created by differing protocol timer values for a single overlay network, with a dynamic link state routing protocol, running on top of a single-domain native topology. The amount of overhead in each layer is different for each scenario. Scenario A has a longer overlay keepAlive-time, but the same native keepAlive-time as Scenario B. As a consequence, the total routing protocol overhead incurred by Scenario B is higher. However, the link failure detection time at the overlay layer is smaller in Scenario B. Our results have shown that it is not necessarily desirable for the overlay layer to detect the failure first. It seems, therefore, that Scenario A is more desirable because

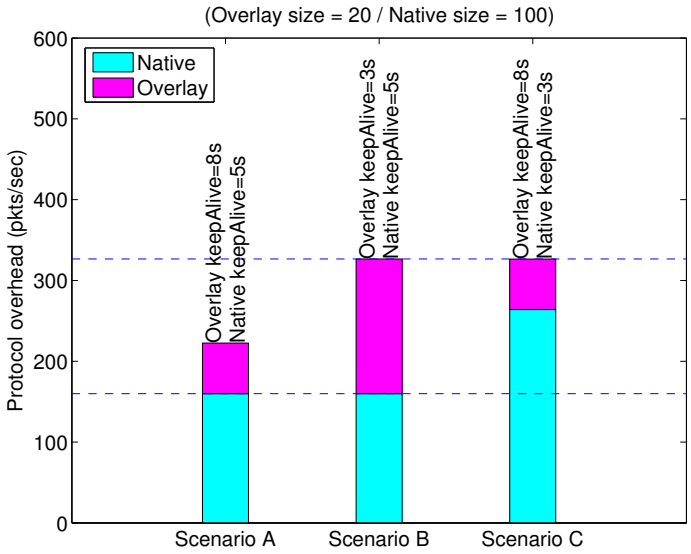


Figure 51: Possible scenarios with multi-layer protocol overhead. Moving from Scenario B to A leads to optimal paths, fewer route flaps and conserves overall routing protocol overhead. Moving from Scenario B to C is what we recommend here.

it insures that the recovery takes place at the native layer first and it incurs less overall protocol overhead.

The assumption that we can change native network parameters, however, makes a third scenario (Scenario C) possible. In this scenario, the overlay keepAlive-time is kept the same as Scenario A, but the native keepAlive-time is decreased so as to trigger an earlier failure detection. The overall routing protocol overhead is the same as Scenario B. But, in Scenario C, we have the desirable property that the native network is able to detect link failures first[3].

The problem of determining the keepAlive-timer values has traditionally been concerned with identifying the right mix of protocol overhead and detection time. We extend the problem by highlighting the need to have the right ratio between native and overlay keepAlive-time, while keeping our solution conservative (same overall protocol overhead and the same effective detection time).

Table 16 presents the relevant results for the three scenarios mentioned (averaged over

[3]There is a possibility that the *keepAlive* messages of the overlay layer are lost due to momentary congestion in the intermediate routers leading to false alarms. But, this does not happen with the native layer as they are exchanged at a higher priority in most networks. Thus, we have an added advantage of lower false alarms with having the native layer detect the failure first.

Table 16: Performance gain with native layer tuning (Topology Size of Overlay = 20, Native = 100)

	Overlay keepAlive	**Native keepAlive**	**Avg. hit-time (s)**	**Avg. num route flap**	**T_{stable}**	**Peak inflation**
A	8	5	12.72	1.571	14.67	1.168
B	3	5	9.166	1.610	11.61	1.247
C	8	3	7.793	1.571	9.739	1.168

25 random single-domain topologies). The native layer tuning we proposed achieves the best performance in all our metrics (viz. hit-time, number of route flaps and path cost inflation), as can be seen from the results in the third row (Scenario C). The value of T_{stable} indicates that the system reaches an earlier stabilization of the paths at the optimal cost. Further, we observe that reducing the keepAlive-time at the native layer also benefits the non-overlay applications sharing it.

B) Tuning the BGP Multi-exit Discriminator for Route Symmetry:

There are many overlay services that require the IP routing to be symmetric[138, 114, 27]. However, this is not true with the current Internet[119]. This can potentially cause the following problems:

- Uneven failure recovery process[138] and complicated troubleshooting, as either direction of the overlay link may not share the same network elements[119].
- Added complexity in having to maintain state information for either direction of an overlay link[114, 27].

Consider the scenario in Fig. 52 where the native layer picks route $KIGDBA$ in one direction and route $ACEFHJ$ in the other. To improve the chances of symmetric routes[4], we need to ensure that the domain uses the same border router for the exit as the entry. As the neighboring AS prefers exit routers with lowest MED value[5], one can tune the BGP MED attribute to pick the required exit router and thereby achieve route symmetry. Since

[4]This does not guarantee an end-to-end symmetric route or a symmetry in link properties.
[5]The default value for the MED attribute in a Cisco 7600 series router is 100.

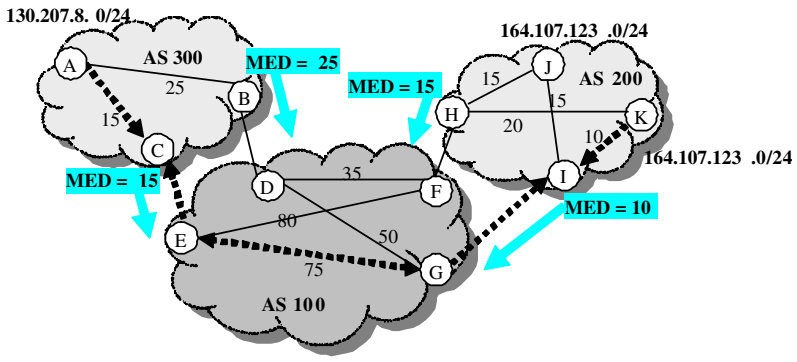


Figure 52: Example to illustrate setting of the MED attribute. The numbers indicated over each iBGP session represent the IGP distance between the two BGP routers. The dashed line indicates the path selected between the two prefixes.

the default scenario of hot-potato routing does not really use the MED value, we do not have any conflict with pre-established BGP policies.

As shown in Fig. 52, when advertising a route to a neighboring AS, the network administrator should set the MED value of each entry point to reflect the relative values of the IGP distance to the entry point. We do this setting only when the AS number of the current domain is bigger than that of the neighboring AS. This causes hot-potato routing in one direction and cold-potato in the other, thereby ensuring piecewise symmetry at the inter-domain level. We can see in Fig. 52 that AS100 prefers to use the exit router G to reach AS200 based on the MED values advertised. In the opposite direction, AS200 uses the closest router I as its exit router to reach AS100, as AS100 did not advertise a particular MED value. Ultimately, the MED tuning causes route $ACEGIK$ to be selected in both directions. This helps the overlay service in question, without harming the non-overlay applications that just look for basic connectivity. The MED tuning inherently requires co-operation between ISPs belonging to different administrative domains and hence brings in the question of incentive, which is out of the scope of this thesis.

C) Tuning IGP Cost for Coherence between Native and Overlay Layers:

A misalignment in the route computation strategy between the native and overlay layers can potentially lead to the following problems[138, 130, 98, 83]:

- Route oscillations that take longer to stabilize
- Defeat of traffic engineering
- Longer overlay paths (in terms of hop count, overall latency etc.)

These problems may also be caused when the two layers adopt mismatching cost schemes. The cost of the individual IGP links at the native layer can be based on hop-count, delay, Euclidean distance, special weights, load or capacity. Similarly, the overlay application might adopt a particular cost scheme based on the type of service offered. These two cost schemes may not necessarily match. For example, an overlay link L_1 with 10 native hops and a delay of 300 ms may be considered longer than link L_2 with 8 native hops and a delay of 400 ms.

In most cases, the overlay application might be able to determine the type of cost metric used by the underlying native layer by means of high-level performance measurements[6]. However, the overlay application might not be able to do that when the cost is based on special weights established by the ISP policy. Such cases will exacerbate the problems due to misalignment. Hence, it is desirable that the native layer IGP costs be configured using easily determinable metrics like propagation delay or hop-count.

7.5 Summary

In this chapter, we stress the need for change at the native layer to suit the evolving overlay services and make the case for deploying an *overlay-friendly native network*. We also present the minimum criteria for designing a native layer change to support a particular overlay service. However, this modification of the native layer may be construed as a contradiction to the reason overlay services were conceived. We argue that while invasive changes are indeed contradictory, other types of changes (such as tuning of the native layer parameters) are non-contradictory.

We consider the evolution of an overlay-friendly native network to be imperative, owing to the high resource usage by the overlay - caused by the potential for widespread use of overlays in the modern Internet, presence of multiple coexisting overlay networks on the

same native topology, and usage of multiple native elements (links and nodes) by each overlay link.

After describing the basic design of the OFNN, we presented examples on tuning existing native layer parameters to improve the performance of the overlay services, as an instance of non-contradictory OFNN design. This form of change at the native layer is non-invasive and is quite easily done by the network administrator. Such a change is not a contradiction in terms. Future work in this direction involves identifying other parameters that need to be tuned.

Though we do not recommend the contradictory OFNN in this chapter, it is perhaps wise not to eliminate it altogether as an option for supporting overlay services. One possible future of the Internet is where overlay services are widely in use and, in such cases, it is easier to justify substantive changes to the native layer. If this does indeed become the case, future work should consider the development of overlay-friendly native network primitives (e.g., the ones proposed by Jannotti[77]) and study the deployability of such modifications. These changes must foster modularity and reusability, so that they can be applicable to a broad range of overlay applications.

We see the following as some of the required follow-up work in this direction:

- Develop more macros that satisfy our design criteria to incorporate in the native layer to help the existing overlay services.
- Designing other overlay services of interest to the consumers, under the assumption that the native layer is willing to cooperate.
- Creating a realistic testbed for these native layer changes. Most testbeds currently available only provide access to the overlay layer and do not yield to a multi-layer test environment.
- Develop ways to prevent a misuse by the overlay layer. When the native layer offers better control of its operation to the overlay layer, there is a chance that the overlay layer might violate the design semantics we specified.

CHAPTER VIII

CONTRIBUTIONS AND FUTURE WORK

Overlay networks offer valuable services needed by end-systems and help overcome functionality limitations of the Internet. However, as shown in our thesis, they lead to complex cross-layer interaction with potentially detrimental effects. We investigated this cross-layer interaction and proposed strategies to achieve a mutually agreeable operating point for both layers. To summarize, the contributions of this thesis are:

- Better control over the cross-layer interaction in the overlay network by incorporating certain simple primitives in either layer. We also show that layer awareness (at the very least an awareness of the existence of the other layer) is crucial to reduce the sub-optimalities. In this context, we propose three different solutions and show that with such schemes one can trade off longer path recovery times with improvements in route flapping and path cost inflation. Further, we show that the best performance is achieved when the native layer can be tuned to suit the overlay needs.

- Insights into the extent and type of inter-domain policy violations caused by service overlay routes. Specifically, we investigate the violation of the commercial agreements between neighboring autonomous systems (AS) and the exit preference of each AS. We also show that the performance of overlay networks will drastically suffer once the native ISPs start enforcing their policies by means of filtering. We propose a cost-sharing approach to mitigate the conflict and to allow the overlay network realize the full advantage of overlay routing without causing native layer policy violations.

- Preemptive game-theoretic strategies that can make each layer improve its performance in a stable manner without requiring any cooperation or information from the other layer. We propose two classes of solutions to improve overlay routing performance: *hostile* strategies that are counterproductive, and *friendly* strategies that

improve performance of both the two layers. This work helps improve coexistence of overlay and native layers, while preparing each other for possible hostile attacks in the future.

- Insights into the selfish behavior and cross-layer interaction in BitTorrent networks. Using real traces and a simulation, we show clear fluctuation in load across inter-domain links. Further, we evaluate existing strategies to reduce the impact of BitTorrent traffic and show that they are ridden with deployability concerns. We propose and evaluate two new BitTorrent strategies, each requiring different levels of cross-layer information, that reduces the detrimental impact of BitTorrent traffic.

- Framework for cross-layer cooperation in the form of overlay-friendliness at the native layer. This overlay-friendly native layer will cater to the overlay applications without compromising on the performance of the non-overlay applications. This friendliness can be achieved by modifying the functioning of the native layer in an invasive or non-invasive manner. We present general design guidelines for this modification and show through examples that we can achieve this friendliness through a non-invasive tuning of the native layer function.

In this thesis, we investigate this tussle as seen in the case of service overlays and BitTorrent applications. We believe that many other forms of selfish interactions can potentially occur when native layer operations (like traffic engineering) are more dynamic and fine-grained. As overlay traffic surges, there is a clear need for our strategies to maintain stability and to improve the performance of each layer with least impact on the other layer.

We learn from this thesis that the overlay layer is quite capable of disrupting the objective of the native layer, and causing instability and deterioration in performance for all applications. Thus, design of the native network without taking into consideration the amount of overlay traffic tends to be futile and causes several impairments. We also learn that layer-awareness is a key requirement for improving end-user performance in the multi-layered Internet. Promoting cross-layer awareness at the overlay layer is essential to overcome existing limitations and to achieve the best possible end-user performance, while

promoting it at the native layer is essential to prepare the ISPs for any negative impact and to achieve the desired performance for all applications.

8.1 Future Work

Besides existing deployment of service overlays, we see that Overlays are essential in the future as:

- Means for end-systems to collaborate and deploy new services
- Environment for testing future innovations (GENI)[62]
- Architecture for future Internet in the form of Network Virtualization[141]

Interestingly, the use of overlay networking in the current Internet, as well as the potential deployments in a future Internet share an important common feature of layering. The "layering" here refers to overlay or virtual networks operating on top of native or substrate networks. In this architecture, cross-layer interaction has a heavy bearing on the achieved routing performance. This urges us to address the following issues:

- Can the performance of end-user applications be further improved? Specifically, we plan to investigate the addition of 1) *Application-awareness* at the network layer, starting with the edge devices in order to incrementally deploy it, 2) *Medium-awareness* (e.g., wireless or wired) at the application layer, so that applications can prepare themselves. This awareness, in turn, requires the addition of layer-to-layer interfaces and common information repositories maintained by the native layer.
- How to prepare ISPs for overlay applications (to promote them or to contain them) effectively? The native network operators are definitely in need of ways to identify overlay traffic and to manage them effectively without causing a deterioration in the user experience for any of their customers. However, identifying overlay traffic is non-trivial, as seen from the unsatisfactory attempts to identify the traffic from Skype. Further, the ISPs are in need of strategies for effectively suppressing malice

from overlay applications and to enforce certain bounds on their functioning, without killing them prematurely.

- How best to tune the network for the new breed of Internet applications and new paradigms of communication? As established by various studies, there is a paradigm shift towards mobile computing and wireless access of services. Furthermore, there has been a shift in the Internet from a *client-server* communication model to a *peer-to-peer* model and now to a *multi-server* model, where multiple servers offer the same content to a large number of pure receivers (freeloaders). This indicates a crucial need for tuning services in the new realm. We need to facilitate this tuning operation effectively, with least impact on legacy traffic.

- How best to design future services in a layer-aware manner? We see that many of the existing services can be offered in either the native or overlay layer. Thus, for each service, we propose to investigate the exact layer where the service must be offered, and the required network level support and cross-layer cooperation in the evolving Internet. The proposed solution must be scalable and easily manageable. Furthermore, each of the services so implemented must be friendly to the existing dynamics of the native layer.

REFERENCES

[1] Aggarwal, V., Feldmann, A., and Scheideler, C., "Can ISPS and P2P users cooperate for improved performance?," *SIGCOMM Comput. Commun. Rev.*, vol. 37, no. 3, pp. 29–40, 2007.

[2] Akella, A., "Understanding the Impact of Route Control Product Deployment," *Invited paper, Workshop on Internet Routing Evolution and Design (WIRED), IEEE Communications Magazine*, October 2003.

[3] Akella, A., Pang, J., Maggs, B., Seshan, S., and Shaikh, A., "A comparison of overlay routing and multihoming route control," in *Proceedings of ACM SIGCOMM*, pp. 93–106, 2004.

[4] Alaettinoglu, C., Jacobson, V., and Yu, H., "Toward millisecond IGP convergence," in *Proceedings of NANOG*, October 2000.

[5] Alaettinoglu, C., Villamizar, C., Gerich, E., Kessens, D., Meyer, D., Bates, T., Karrenberg, D., and Terpstra, M., "Routing policy specification language (RPSL)." Request for Comments 2622, June 1999.

[6] Andersen, D., Balakrishnan, H., Kaashoek, M. F., and Morris, R., "Resilient Overlay Networks," in *Proceedings of 18th ACM SOSP*, October 2001.

[7] Autenrieth, A., Doorselaere, K., and Demeester, P., "Evaluation of multilayer recovery strategies," in *Proceeding of Eunice '98 Open*, pp. 33–42, September 1998.

[8] "Bad ISPs." http://azureuswiki.com/index.php/Bad_ISPs.

[9] Barford, P., Banerjee, S., and Estan, C., "Design for Manageability in the Next Generation Internet." National Science Foundation NeTS-FIND Sponsored Program, October 2006.

[10] Barnako, F., "BellSouth wants new Net fees," *Market Watch*, vol. `http://www.marketwatch.com/news/story.asp?guid=\%7B02432D2D-1EE0-4%037-A15F-54B748D6CF26\%7D`, January 2006.

[11] Baset, S. A. and Schulzrinne, H., "An Analysis of the Skype Peer-to-Peer Internet Telephony Protocol," Tech. Rep. CUCS-039-04, Columbia University, New York, September 2004.

[12] Bavier, A., Feamster, N., Huang, M., Peterson, L., and Rexford, J., "In VINI veritas: realistic and controlled network experimentation," in *Proceedings of ACM SIGCOMM*, pp. 3–14, 2006.

[13] Bavier, A., Huang, M., and Peterson, L., "An Overlay Data Plane for PlanetLab," in *Proceedings of the Advanced Industrial Conference on Telecommunications*, July 2005.

[14] Bharambe, A., Herley, C., and Padmanabhan, V., "Some observations on bittorrent performance," in *Proceedings of ACM SIGMETRICS*, 2005.

[15] Bharambe, A., Herley, C., and Padmanabhan, V., "Analyzing and Improving a BitTorrent Network's Performance Mechanisms," in *Proceedings of IEEE INFOCOM*, April 2006.

[16] "Bittorrent." `http://www.bittorrent.com`.

[17] Blanton, E. and Allman, M., "On making TCP more robust to packet reordering," *ACM Computer Communication Review*, January 2002.

[18] Braynard, R., Kostic, D., Rodriguez, A., Chase, J., and Vahdat, A., "Opus: an Overlay Peer Utility Service," in *Proceedings of the 5th OPENARCH*, June 2002.

[19] "BitTorrent lands new hardware deals." CNET News.com, `http://news.com.com/2100-1025_3-6128583.html`, October 2006.

[20] "Message Stream Encryption." `http://azureuswiki.com/index.php/Message_Stream_Encryption`.

[21] "Cachelogic's VelociX Network." `http://www.cachelogic.com`.

[22] "Peer-to-Peer in 2005." Cachelogic White Paper, `http://www.cachelogic.com/home/pages/research/p2p2005.php`.

[23] Calvert, K., Doar, M., and Zegura, E., "Modeling Internet Topology," *IEEE Communications Magazine*, June 1997.

[24] Chandrashekar, J., Zhang, Z., Duan, Z., and Hou, Y. T., "Service Oriented Internet," in *Proceedings of the 1st ICSOC*, December 2003.

[25] Chawathe, Y., "Scattercast: an adaptable broadcast distribution framework," *Multimedia Systems*, pp. 104–118, July 2003.

[26] Chen, S. and Nahrstedt, K., "An Overview of Quality-of-Service Routing for the Next Generation High-Speed Networks: Problems and Solutions," *IEEE Network Magazine*, November 1998.

[27] Chen, Y., Bindel, D., Song, H., and Katz, R., "An Algebraic Approach to Practical and Scalable Overlay Network Monitoring," in *Proceedings of ACM SIGCOMM*, 2004.

[28] Chu, Y., Rao, S., and Zhang, H., "A Case for End System Multicast," in *Proceedings of ACM SIGMETRICS*, June 1999.

[29] Clark, D., Lehr, W., Bauer, S., Faratin, P., Sami, R., and Wroclawski, J., "The Growth of Internet Overlay Networks: Implications for Architecture, Industry Structure and Policy," in *Proceedings of Telecommunications Policy Research Conference (TPRC-05)*, September 2005.

[30] Clímaco, J. and Martins, E., "A bicriterion shortest path algorithm," *European Journal of Operational Research*, vol. 11, pp. 399–404, 1982.

[31] Cohen, B., "Incentives Build Robustness in BitTorrent," in *Proceedings of the 1st Workshop on Economics of Peer-to-Peer Systems*, June 2003.

[32] Cohen, B., "Obfuscating BitTorrent." `http://bramcohen.livejournal.com/29886.html`, January 2006.

[33] Corley, H. and Moon, I., "Shortest paths in networks with vector weights," *Journal of Optimization Theory and Application*, vol. 46, pp. 79–86, May 1985.

[34] Cui, W., Stoica, I., , and Katz, R., "Backup Path Allocation based on a Link Failure Probability Model in Overlay Networks," in *Proceedings of ICNP*, November 2002.

[35] Davie, B. and Rekhter, Y., *MPLS: Technology and Applications*. San Francisco, CA, Morgan Kaufmann, 2000.

[36] Dimitropoulos, X. and Riley, G., "Creating Realistic BGP Models," in *Proceedings of 11th IEEE/ACM International Symposium on Modeling*, October 2003.

[37] "Overcoming Barriers to Disruptive Innovation in Networking." NSF workshop report, http://www.planet-lab.org/doc/barriers.pdf, Jan 2005.

[38] Dovrolis, C., Ramanathan, P., and Moore, D., "What Do Packet Dispersion Techniques Measure?," in *Proceedings of IEEE INFOCOM*, pp. 905–914, 2001.

[39] Duany, Z., Zhangy, Z., and Houz, Y., "Service Overlay Networks: SLAs, QoS and Bandwidth Provisioning," in *Proceedings of ICNP*, Nov 2002.

[40] E., S. R., *Multiple Criteria Optimization: Theory, Computation and Application*. Wiley, 1986.

[41] "eDonkey2000 - Overnet." `http://web.archive.org/web/20060601102419/http://www.edonkey.com/`.

[42] Ehrgott, M., *Multicriteria optimization*. Lecture Notes in Economics and Mathematical Systems, Springer-Verlag, 2000.

[43] "Web traffic overtakes peer-to-peer (p2p) in broadband usage in north america." `http://www.ellacoya.com/news/pdf/2007/NXTcommEllacoyaMediaAlert.pdf`.

[44] Elwalid, A., Jin, C., Low, S., and Widjaja, I., "MATE: MPLS Adaptive Traffic Engineering," in *Proceedings of IEEE INFOCOM*, 2001.

[45] "Emulab." `http://www.emulab.org`.

[46] et al., B. Q., "Interdomain traffic engineering with BGP," *IEEE Communications Magazine*, 2003.

[47] et al., M. I., "Dissecting BitTorrent: Five months in a torrent's lifetime," in *Proceedings of the 5th PAM Workshop*, April 2004.

[48] et al., M. P., "Do incentives build robustness in BitTorrent?," in *4th USENIX Symposium on NSDI*, April 2007.

[49] ET AL., P. D., "PANEL - Protection across network layers," in *Proceeding of European Conference on Networks and Optical Communications*, June 1997.

[50] ET AL., R. B., "Improving Traffic Locality in BitTorrent via Biased Neighbor Selection," in *Proceedings of the 26th IEEE International Conference on Distributed Computing Systems (ICDCS)*, 2006.

[51] ET AL., S. S., "Detour: a Case for Informed Internet Routing and Transport," Tech. Rep. TR-98-10-05, U. of Washington, Seattle, 1998.

[52] FALK, A., "Not quite the differentiated services I was thinking of." Note sent to e2e mailing list, October 2005.

[53] FEAMSTER, N. and L. GAO, J. R., "CABO: Concurrent Architectures are Better than One." National Science Foundation NeTS-FIND Sponsored Program, September 2006.

[54] FLOYD, S., ALLMAN, M., JAIN, A., and SAROLAHTI, P., "Quick-Start for TCP and IP." RFC 4782, January 2007.

[55] FORTZ, B. and THORUP, M., "Internet Traffic Engineering by Optimizing OSPF Weights," in *INFOCOM*, pp. 519–528, 2000.

[56] FREEMAN, L., BORGATTI, S., and WHITE, D., "Centrality in Valued Graphs: A Measure of Betweeness Based on Network Flow," *Social Networks*, vol. 13, no. 141, pp. 141–154, 1991.

[57] FUDENBERG, D. and TIROLE, J., *Game Theory*. MIT Press, 1991.

[58] FUMAGALLI, A. and VALCARENGHI, L., "IP Restoration vs. WDM Protection: Is There an Optimal Choice?," *IEEE Network Magazine*, pp. 34–41, November 2000.

[59] GAO, L., "On inferring autonomous system relationships in the internet," *IEEE/ACM Trans. Netw.*, vol. 9, no. 6, pp. 733–745, 2001.

[60] GAREY, M. R. and JOHNSON, D. S., *Computer and Intractability: A Guide to the Theory of NP-Completeness.* W. H. Freeman, 1979.

[61] GEIST, M., "Towards a two-tier internet," *BBC News*, vol. `http://news.bbc.co.uk/1/hi/technology/4552138.stm`, December 2005.

[62] "GENI: Global Environment for Network Innovations." `http://www.geni.net`.

[63] GKANTSIDIS, C., MILLER, J., and RODRIGUEZ, P., "Anatomy of a P2P Content Distribution System with Network Coding," in *IPTPS*, 2006.

[64] GKANTSIDIS, C. and RODRIGUEZ, P., "Network Coding for Large Scale Content Distribution," in *Proceedings of IEEE INFOCOM*, 2005.

[65] "GNU Linear Programming Kit (GLPK)." `http://www.gnu.org/software/glpk`.

[66] GOYAL, M., RAMAKRISHNAN, K., and FENG, W., "Achieving Faster Failure Detection in OSPF Networks," in *Proceedings of ICC*, 2003.

[67] Griffin, T. and Wilfong, G. T., "Analysis of the MED Oscillation Problem in BGP," in *Proceedings of ICNP*, pp. 90–99, November 2002.

[68] Guo, L., Chen, S., Xiao, Z., Tan, E., Ding, X., and Zhang, X., "Measurements, analysis, and modeling of bittorrent-like systems," in *Proceedings of ACM IMC*, pp. 35–48, 2005.

[69] Han, J., Watson, D., and Jahanian, F., "Topology Aware Overlay Networks," in *Proceedings of IEEE INFOCOM*, Mar 2005.

[70] Handley, M., Padhye, J., and Floyd, S., "TCP Congestion Window Validation." RFC 2861, June 2000.

[71] Handley, M., Kohler, E., Ghosh, A., Hodson, O., and Radoslavov, P., "Designing Extensible IP Router Software," in *Proceedings of the 2nd USENIX Symposium on Networked Systems Design and Implementation (NSDI '05)*, (Boston, MA, USA), May 2005.

[72] Hansen, P., "Bicriterion path problems," in *Multiple Criteria Decision Making: Theory and Applications, LNEMS 177*, pp. 109–127, Springer-Verlag, Berlin, 1980.

[73] Hochbaum, D., *Approximation Algorihtms for NP-Hard Problems*. Brooks/Cole Publishing Co., 1996.

[74] "The Continuing Boom of P2P File Sharing." `http://www.ipoque.com/en/pressrelease_ipoque_241006.html`, October 2006.

[75] "Internet Routing Registry." `http://www.irr.net/docs/list.html`.

[76] Jain, M. and Dovrolis, C., "Pathload: A measurement tool for end-to-end available bandwidth," in *Proceedings of the PAM Workshop*, March 2002.

[77] Jannotti, J., *Network layer support for overlay networks*. PhD thesis, Massachusetts Institute of Technology, 2002. Supervisor-M. Frans Kaashoek.

[78] Jannotti, J., Gifford, D., Johnson, K., Kaashoek, F., and O'Toole, J., "Overcast: Reliable Multicasting with an Overlay Network," in *4th USENIX OSDI Symposium*, October 2000.

[79] Kandula, S., Katabi, D., Davie, B., and Charny, A., "Walking the Tightrope: Responsive Yet Stable Traffic Engineering," in *Proceedings of ACM SIGCOMM*, August 2005.

[80] Karagiannis, T., Rodriguez, P., and Papagiannaki, K., "Should Internet Service Providers Fear Peer-Assisted Content Distribution?," in *Proceedings of ACM Internet Measurement Conference*, Otober 2005.

[81] Karagiannis, T., Papagiannaki, K., and Faloutsos, M., "BLINC: multilevel traffic classification in the dark," in *Proceedings of ACM SIGCOMM*, pp. 229–240, 2005.

[82] Karbhari, P., Ammar, M., and Zegura, E., "Optimizing End-to-End Throughput for Data Transfers on an Overlay-TCP Path," in *Proceedings of Networking 2005*, May 2005.

[83] Keralapura, R., Taft, N., Chuah, C. N., and Iannaccone, G., "Can ISPs take the heat from Overlay Networks?," in *Proceedings of ACM HotNets-III*, November 2004.

[84] Keralapura, R., Taft, N., Chuah, C. N., and Iannaccone, G., "Can coexisting overlays inadvertently step on each other?," in *Proceedings of IEEE ICNP*, November 2005.

[85] Keromytis, A., Misra, V., and Rubenstein, D., "SOS: Secure Overlay Services," in *Proceedings of ACM SIGCOMM*, 2002.

[86] Kodialam, M. and Lakshman, T. V., "Dynamic routing of bandwidth guaranteed tunnels with restoration," in *Proceedings of IEEE INFOCOM*, March 2000.

[87] Korilis, Y., Lazar, A., and Orda, A., "Achieveing Network Optima using Stackelberg Routing Strategies," *IEEE/ACM Transactions on Networking*, vol. 5, no. 1, pp. 161–173, 1997.

[88] Koutsoupias, E. and Papadimitriou, C., "Worst-case Equilibria," in *Proceedings of the 16th Annual Symposium on Theoretical Aspects of Computer Science*, pp. 404–413, 1999.

[89] Krim, J., "Executive Wants to Charge for Web Speed," *Washington Post*, vol. `http://www.washingtonpost.com/wp-dyn/content/article/2005/11/30/AR%2005113002109.html`, December 2005.

[90] Kuhns, F., Wilson, M., and Turner, J., "Diversifying the Network Edge," in *Work in progress*, 2005.

[91] Kwon, M. and Fahmy, S., "Toward Cooperative Inter-overlay Networks," in *Proceedings of ICNP*, November 2003.

[92] Labovitz, C., Ahuja, A., Bose, A., and Jahanian, F., "Delayed Internet routing convergence," *IEEE/ACM Transactions on Networking*, pp. 293–306, June 2001.

[93] Legout, A., Urvoy-Keller, G., and Michiardi, P., "Understanding BitTorrent: An Experimental Perspective," Tech. Rep. 00000156, Version 2, INRIA, July 2005.

[94] Legout, A., Urvoy-Keller, G., and Michiardi, P., "Rarest First and Choke Algorithms Are Enough," in *Proceedings of the Internet Measurement Conference (IMC)*, October 2006.

[95] Li, Z. and Mohapatra, P., "The Impact of Topology on Overlay Routing Service," in *Proceedings of IEEE INFOCOM*, March 2004.

[96] Li, Z., Mohapatra, P., and Chuah, C. N., "Virtual Multihoming: On the Feasibility of Combining Overlay Routing with BGP Routing," in *Proceedings of Networking*, 2005.

[97] "Complex Deployment and Analysis of Link-state Protocols." Cisco Press, Networkers, 2003.

[98] Liu, Y., Zhang, H., Gong, W., and Towsley, D., "On the Interaction Between Overlay Routing and Traffic Engineering," in *Proceedings of IEEE INFOCOM*, 2005.

[99] "Looking Glass servers." http://www.traceroute.org.

[100] L.Qiu, R.Y.Yang, Y.Zhang, and Shenker, S., "On Selfish Routing in Internet-Like Environments," in *Proceedings of ACM SIGCOMM*, 2003.

[101] Lu, J. and Turner, J., "Efficient Mapping of Virtual Network onto a Shared Substrate," tech. rep., Washington University in St. Louis, June 2006.

[102] Mahajan, R., Wetherall, D., and Anderson, T., "Understanding BGP misconfiguration," in *Proceedings of ACM SIGCOMM*, pp. 3–16, 2002.

[103] Mao, Z., Johnson, D., Rexford, J., Wang, J., and Katz, R., "Scalable and accurate identification of AS-level forwarding paths," in *Proceedings of IEEE INFOCOM*, March 2004.

[104] Mao, Z. M., Qiu, L., Wang, J., and Zhang, Y., "On AS-level path inference," in *Proceedings of ACM SIGMETRICS*, pp. 339–349, 2005.

[105] Markopoulou, A., Iannaccone, G., Bhattacharyya, S., Chuah, C.-N., and Diot, C., "Characterization of Failures in an IP Backbone," in *Proceedings of IEEE INFOCOM*, April 2004.

[106] Martins, E., "On a multicriteria shortest path problem," *European Journal of Operational Research*, vol. 16, pp. 236–245, 1984.

[107] Medina, A., Taft, N., Battacharya, S., Diot, C., and Salamatian, K., "Traffic matrix estimation: Existing techniques and new directions," in *Proceedings of ACM SIGCOMM*, 2002.

[108] Mier, E., Mier, D., and Mosco, A., "Assessing Skype's network impact," *Network World*, vol. http://www.networkworld.com/reviews/2005/121205-skype-test.html, December 2005.

[109] Mohan, G., Murthy, C. S. R., and Somani, A., "Efficient algorithms for routing dependable connections in WDM optical networks," *IEEE/ACM Transactions on Networking*, pp. 553–566, October 2001.

[110] Morris, R., "TCP behavior with many flows," in *Proceedings of the International Conference on Network Protocols (ICNP)*, p. 205, 1997.

[111] Moy, J. T., *Ospf: Anatomy of An Internet Routing Protocol*. Addison Wesley Professional, 1998.

[112] "Multiprotocol label switching (MPLS)." http://www.ietf.org/html.charters/mpls-charter.html.

[113] Mysore, J. and Bharghavan, V., "A new multicasting-based architecture for Internet host mobility," in *Proceedings of ACM/IEEE MobiCom '97*, pp. 161–172, September 1997.

[114] Nakao, A., Peterson, L., and Bavier, A., "A routing underlay for overlay networks," in *Proceedings of ACM SIGCOMM*, August 2003.

[115] Norton, W. B., "A Business Case for ISP Peering." White Paper, http://www.equinix.com, February 2002.

[116] "Orbitlab." `http://www.orbit-lab.org`.

[117] "Packeteer's PacketShaper." `http://www.packeteer.com/prod-sol/solutions/p2p.cfm`.

[118] Padhye, J., Firoiu, V., Towsley, D., and Kurose, J., "Modeling TCP Throughput: A Simple Model and its Empirical Validation," *Proceedings of ACM SIGCOMM*, pp. 303–314, 1998.

[119] Paxson, V., "End-to-end Internet packet dynamics," *IEEE/ACM Transactions on Networking*, pp. 277–292, 1997.

[120] Peterson, L., Culler, D., Anderson, T., and Roscoe, T., "A Blueprint for Introducing Disruptive Technology into the Internet," in *Proceedings of ACM HotNets-I*, October 2002.

[121] "Planetlab." `http://www.planet-lab.org`.

[122] Poretsky, S., "Considerations for Benchmarking IGP Data Plane Route Convergence." Work in progress, Internet Draft draft-ietf-bmwg-igp-dataplane-conv-app-05.txt, February 2005.

[123] Pouwelse, J., Garbacki, P., Epema, D., and Sips, H., "The Bittorrent P2P File-sharing System: Measurements and Analysis," in *IPTPS*, 2005.

[124] Pouwelse, J. A., Garbacki, P., Epema, D. H. J., and Sips, H. J., "The Bittorrent P2P File-sharing System: Measurements and Analysis," in *4th Int'l Workshop on Peer-to-Peer Systems (IPTPS)*, Feb 2005.

[125] Qiu, D. and Srikant, R., "Modeling and performance analysis of BitTorrent-like peer-to-peer networks," in *Proceedings of ACM SIGCOMM*, pp. 367–378, 2004.

[126] Reed, D., Saltzer, J., and Clark, D., "Active Networking and End-to-End Arguments," *IEEE Network*, pp. 67–71, May 1998.

[127] Rekhter, Y. and Li, T., "A Border Gateway Protocol 4 (BGP-4)." RFC 1771, March 1995.

[128] "RIPE Routing Information Services." `http://ripe.net/ris`.

[129] Roscoe, T., "The End of Internet Architecture," in *Proceedings of ACM HotNets-V*, November 2006.

[130] Roughgarden, T. and Tardos, E., "How Bad is Selfish Routing?," *Journal of the ACM*, vol. 49(2), pp. 236–259, March 2002.

[131] "Route Views Project." `http://www.routeviews.org/`.

[132] Sahasrabuddhe, L., Ramamurthy, S., and Mukherjee, B., "Fault Management in IP-Over-WDM Networks: WDM Protection vs. IP Restoration," *IEEE Journal on Selected Areas in Communications*, January 2002.

[133] Saltzer, J., Reed, D., and Clark, D., "End-To-End Arguments in System Design," *ACM Transactions on Computer Systems*, 1984.

[134] "Sandvine's PPE 8200." http://www.sandvine.com/products/p2p_element.asp.

[135] Schroeder, B. and Harchol-Balter, M., "Evaluation of task assignment policies for supercomputing servers: The case for load unbalancing and fairness," in *Proceedings of HPDC*, 2000.

[136] Seetharaman, S. and Ammar, M., "Overlay-friendly Native Network: A Contradiction in Terms?," in *Proceedings of ACM HotNets-IV*, November 2005.

[137] Seetharaman, S. and Ammar, M., "Characterizing and Mitigating Inter-domain Policy Violations in Overlay Routes," in *Proceedings of IEEE ICNP*, November 2006.

[138] Seetharaman, S. and Ammar, M., "On the Interaction between Dynamic Routing in the Overlay and Native Layers," in *Proceedings of IEEE INFOCOM*, April 2006.

[139] Seetharaman, S., Hilt, V., Hofmann, M., and Ammar, M., "Preemptive Strategies to Improve Routing Performance of Native and Overlay Layers," in *Proceedings of IEEE INFOCOM*, May 2007.

[140] Sen, S. and Wang, J., "Analyzing peer-to-peer traffic across large networks," in *Second Annual ACM Internet Measurement Workshop*, November 2002.

[141] Shenker, S., Peterson, L., and Turner, J., "Overcoming the Internet Impasse through Virtualization," in *Proceedings of ACM HotNets-III*, November 2004.

[142] Shi, S. and Turner, J., "Placing Servers in Overlay Networks," *SPECTS*, July 2002.

[143] Simpson, W., "IP in IP tunneling." RFC 1853, October 1995.

[144] "SonicWALL's Unified Threat Management." `http://www.sonicwall.com/products/utm.html`.

[145] Spring, N., Wetherall, D., and Anderson, T., "Scriptroute: A facility for distributed Internet measurement," in *4th USENIX Symposium on Internet Technologies and Systems*, March 2003.

[146] Spring, N., Mahajan, R., and Wetherall, D., "Measuring ISP topologies with Rocketfuel," in *Proceedings of ACM SIGCOMM*, 2002.

[147] Stoica, I., Adkins, D., Zhuang, S., Shenker, S., and Surana, S., "Internet Indirection Infrastructure," in *Proceedings of ACM SIGCOMM*, August 2002.

[148] Stone, B., "Software Exploited by Pirates Goes to Work for Hollywood," *NY Times*, vol. `http://www.nytimes.com/2007/02/25/technology/25bit.html`, February 2007.

[149] Subramanian, L., Stoica, I., Balakrishnan, H., and Katz, R., "OverQoS: offering Internet QoS using overlays," in *Proceedings of ACM SIGCOMM*, August 2003.

[150] Suh, K., Figueiredo, D., Kurose, J. F., and Towsley, D., "Characterizing and detecting relayed traffic: A case study using Skype," in *Proceedings of IEEE INFOCOM*, April 2006.

[151] Tangmunarunkit, H., Govindan, R., Shenker, S., and Estrin, D., "The Impact of Routing Policy on Internet Paths," in *Proceedings of IEEE INFOCOM*, pp. 736–742, 2001.

[152] Taylor, D. and Turner, J., "Towards a Diversified Internet." White paper, November 2004.

[153] Tennenhouse, D., Smith, J., Sincoskie, W. D., Wetherall, D., and Minden, G., "A Survey of Active Network Research," *IEEE Communications Magazine*, vol. 35, January 1997.

[154] Theimer, M. and Jones, M. B., "Overlook: Scalable Name Service on an Overlay Network," in *Proceedings of the 22nd International Conference on Distributed Computing Systems*, July 2002.

[155] Touch, J., "Dynamic Internet Overlay Deployment and Management Using the X-Bone," *Computer Networks*, pp. 117–135, July 2001.

[156] Touch, J., Finn, G., Wang, Y., and Eggert, L., "DynaBone: dynamic defense using multi-layer Internet overlays," in *Proceedings of DARPA Information Survivability Conference and Exposition*, pp. 271–276, April 2003.

[157] Touch, J., Y., Y. W., and Pingali, V., "A Recursive Network Architecture," Tech. Rep. ISI-TR-2006-626, Information Sciences Institute, October 2006.

[158] Turner, J. and Taylor, D., "Diversifying the Internet," in *Proceedings of IEEE GLOBECOM*, 2005.

[159] Valiant, L. G., "A scheme for fast parallel communication," *SIAM Journal on Computing*, vol. 11, no. 2, pp. 350–361, 1982.

[160] Vasseur, J., Pickavet, M., and Demeester, P., *Network recovery: Protection and Restoration of Optical, SONET-SDH, IP, and MPLS*. Morgan Kaufmann, July 2004.

[161] "Verso Technologies' NetSpective P2PFilter." `http://www.verso.com/enterprise/netspective/p2pfilter.asp`.

[162] Wang, H., Xie, H., Qiu, L., Yang, R., Zhang, Y., and Greenberg, A., "COPE: Traffic Engineering in Dynamic Networks," in *Proceedings of ACM SIGCOMM*, September 2006.

[163] Warburton, A., "Approximation of Pareto Optima in Multiple-Objective Shortest-Path Problems," *Operations Research*, vol. 35, pp. 70–79, 1987.

[164] Wu, C. and Li, B., "Strategies of Conflict in Coexisting Streaming Overlays," in *Proceedings of IEEE INFOCOM*, May 2007.

[165] Wun, B., Turner, J., and Crowley, P., "Virtualizing Network Processors," tech. rep., Washington University in St. Louis, March 2006.

[166] Xia, J. and Gao, L., "On the evaluation of AS relationship inferences," in *Proceedings of IEEE GLOBECOM*, December 2004.

[167] Xie, H., Krishnamurthy, A., Silberschatz, A., and Yang, Y. R., "P4P: Explicit Communications for Cooperative Control Between P2P and Network Providers." `http://www.dcia.info/documents/P4P_Overview.pdf`.

[168] Xiong, Y. and Mason, L., "Restoration Strategies and Spare Capacity Requirements in Self-healing ATM Networks," *IEEE/ACM Transactions on Networking*, pp. 98–110, February 1999.

[169] Yalagandula, P., Sharma, P., Banerjee, S., Basu, S., and Lee, S.-J., "S3 : A Scalable Sensing Service for Monitoring Large Networked Systems," in *Proceedings of the Internet Network Management (INM'06) workshop*, September 2006.

[170] Yang, X. and de Veciana, G., "Service Capacity of Peer to Peer Networks," in *Proceedings of IEEE INFOCOM*, 2004.

[171] "Youttube - broadcast yourself." `http://www.youtube.com`.

[172] Zhang, B., Liu, R., Massey, D., and Zhang, L., "Collecting the Internet AS-level topology," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 35, no. 1, pp. 53–61, 2005.

[173] Zhang, H., Liu, Y., Gong, W., and Towsley, D., "Understanding the interaction between overlay routing and MPLS traffic engineering," Tech. Rep. UM-CS-2004-063, University of Mass. at Amherst, 2004.

[174] Zhang, H., Kurose, J. F., and Towsley, D., "Can an overlay compensate for a careless underlay?," in *Proceedings of IEEE INFOCOM*, April 2006.

[175] Zhao, B., Duan, Y., Huang, L., Joseph, A., and Kubiatowicz, J., "Brocade: Landmark routing on overlay networks," in *1st International Workshop on Peer-to-Peer Systems (IPTPS)*, March 2002.

[176] Zhu, Y., Dovrolis, C., and Ammar, M., "Dynamic Overlay Routing Based on Available Bandwidth Estimation: A Simulation Study," *Computer Networks Journal (Elsevier)*, vol. 50, pp. 739–876, April 2006.

[177] Zhu, Y. and Ammar, M., "Algorithms for Assigning Substrate Network Resources to Virtual Network Components," in *Proceedings of IEEE INFOCOM*, 2006.

[178] Zhu, Y. and Ammar, M., "Overlay network assignment in PlanetLab with NetFinder," Tech. Rep. GT-CSS-06-11, College of Computing, Georgia Institute of Technology, August 2006.

[179] Zhuang, S., Geels, D., Stoica, I., and Katz, R., "On Failure Detection Algorithms in Overlay Networks," in *Proceedings of INFOCOM*, 2005.